**Interruption-Information Management Framework for Chat Interface**

Shreejana Prajapati, Koichi Yamada, Muneyuki Unehara, and Izumi Suzuki

Graduate School of Engineering, Nagaoka University of Technology, 1603-1 Kamitomioka-Machi, Nagaoka, Niigata 940-2188, Japan
Email: s105087@stn.nagaokaut.ac.jp, {yamada, unehara, suzuki}@kjs.nagaokaut.ac.jp

**Abstract**—Spontaneous communication is an integral part of any workplace as well as everyday life. Most of the spontaneous communications happen over email or chat applications in workplaces that use computer technologies. Frequent use of chat application or email hampers the recipient’s workflow and leads to constant interruptions causing task fragmentation. In this paper, we present a receiver oriented interruption-information management framework using automated agents to prevent receivers from a plethora of messages. This framework manages both the interruption and upcoming information in the chat interface. This study is a novel approach in the area of interruption management, which not only considers interruption management but also processes information based on users’ behavior and preferences. This framework is a cooperative approach where both the message sender and the receiver work together to deliver messages during favorable times. The receiver contributes to manage interruption whereas the sender helps to manage information, together forming an interruption-information management mechanism for deciding the least interruptible time for message delivery. The study shows that users become comparatively more productivity and complete more task with the relatively short amount of time while using the interruption-information management chat interface than conventional chat.

I. INTRODUCTION

Timely update of new information and task attainment are positive aspects of chat interfaces. However, interruptions caused by unwanted messages at inappropriate timing are their unavoidable drawback, and such computer technologies are considered to exacerbate problems [9]. We present a novel approach of interruption-information management to overcome this drawback. Our approach handles message overflow and ensures appropriate responsiveness of chat interfaces. Despite of emerging studies on interruption management and its drawbacks, none of them have studied both interruption and information management in the chat application. They mostly focused only on the aspects of interruptions.

Chat interfaces are designed specifically for prompt message delivery. If we incorporate a management system focusing only on preventing interruption, they will hamper their designed purpose. Instead, we aim for interruption-information management particularly for chat interfaces taking into account preferences of both senders and receivers. This interruption-information management framework prevents chat interfaces from being an interrupter and also ascertain timely delivery of messages. In this framework, interactions involve individuals with conflicting objectives, where one demands prompt message delivery while the other tries to delay unnecessary frequent messages to prevent interruption. This framework is designated to work in situations where a single sender communicates with a single receiver as well as multiple senders communicate with a single receiver. When multiple senders want to start a chat communication with the same single receiver, the receiver must place the priority order and prioritize one of them. We have incorporated this idea in to our framework where all the senders need to compete with each other for the earliest possible message delivery. This is also an improvement over existing chat applications as we introduce a competitive approach in multiple senders to the same receiver scenario.

Our previous study [2] suggests Self-Initiated Intermission (SI) is a better way for handling interruptions. Any message sent without checking the contextual appropriateness has more chances of becoming an interruption. Hence, if a message is delivered during the receiver’s intermission, it has a high probability of being read and would not cause interruption to the receiver. We established a concept of delivering messages only at a time of intermission. However, this will not be feasible when the receiver does not take intermissions for a long duration, which causes an urgent message to be queued to wait for an intermission and discourages the message senders. So, we present a concept of cooperation between software agents for appropriate time when intermission is not feasible.

II. RELATED WORK

The study by Shiu and Lenhart [4] shows that the use of Instant Messenger (IM) is increasing at the rate of 9% per annum. Along with its proliferation from personal to formal organizational use [5], it has attracted researchers’ attention since IM forces users to switch from their current work causing interruption. It affects users’ task performance [5], [10] as the resumption lag [16], i.e., time taken to resume their work after the interruption is usually longer than expected [13]. The relation between interruption (message) frequency, task complexity and user satisfaction showed polychrons are
comparatively satisfied with their work progress than the monochrons [11]. Monochrons prefer working on one task at a time and become disoriented in a multitasking environment. Whereas, polychrons prefers multitasking environment. The same study demonstrated that polychrons deal with all kinds of messages in the same manner irrespective of the hierarchy of the sender. Our proposed framework for the interruption and information management is to facilitate both the monochrons and the polychrons to achieve work satisfactions.

Despite the interruption frequency, interruption management is essential to satisfy users with their work progress. Tanaka and Fujita [8] presented a mediated way of interruption coordination using a secretary agent to inform users about upcoming interruptions. In addition, Tankaka et. al. [12] presented interruption estimation based on the motion of head tested on a single task. The frequency of the head motion will be comparatively higher in a multitasking environment and most of the time it might be due to external factors, so head motion does not give a promising interruption estimation.

NegChat [1] is a chat based negotiating agent that uses Aspiration Adaptation Theory (AAT) to negotiate with the human to reach an agreement. AAT [7] gives the concept that human negotiator uses bounded rationality to attain the agreement [1], [3], [6]. The result showed that negotiating agents were more effective and got through more agreements than human negotiators. However, in the research based on AAT, if the agreement cannot be made, then the agent or the negotiating party lowers his/her value for a good enough deal.

In our framework, involving parties cooperates with each other to achieve an agreement that is the best for both.

Existing literature, however, have studied agreement made by chat based agent, but agreement based on controlled interruption for chat interface is still untouched. This is a pioneer work on the field of interruption management that handles interruptions and also manage information in chat interface. In the hierarchy of communication media, IM is a richer medium than Email [14], based on their capacity to convey information [15]. So, information management is incorporated in this framework to maintain the designed purpose of IM.

In this study, we present a framework acknowledging the priorities of both involving parties for the better interruption and information management. Our approach is different from other interruption management work as it considers users’ behavior to control both interruption and information. It is a scheduled and cooperative way of management between senders and a receiver which is carried out by automated agents. We recommend this interruption-information management framework to facilitate controlled message delivery with acceptable deferral based on urgency, i.e., delaying less urgent messages and minimizing interruptions. This framework is new and manages interruption and information with the users’ preferences.

III. INTERRUPTION-INFORMATION MANAGEMENT

The interruption-information management framework for a chat interface consists of a receiver and a single/multiple senders who control interruption and manage information such that important messages can be delivered with an acceptable delay whereas lower priority messages wait for a favorable time. This framework concentrates on the chat interface struggling to manage a bombardment of messages. In existing messaging systems, senders send messages without considering receiver’s inconveniences and interruptions. Therefore, we embed the concept of using agents to facilitate both the sender and the receiver for interruption and information management based on their preferences. It also provides the facility to select the intermission mode when users want to take a break, which indicates that any interruption is acceptable.

This framework provides an automated interruption-information management where the senders’ and receivers’ role is just to send and receive the messages. The respective agents do the rest of the process such as holding, prioritizing and choosing an appropriate time to deliver the message. The sender agents contribute to some portion of interruption-information management by managing information based on message priority, while the receiver agent contributes to the rest by managing interruption based on sender-receiver relationship and message waiting time. In typical chat applications, senders’ the main objective is to deliver a message as soon as possible, regardless of the urgency of the message as well as the receivers’ preferences. However, this framework is designed to cooperate with each other for less interruptive message delivery with a minimum delay. We further explain the individual components and the cooperation between the sender agents and the receiver agent regarding interruption and information management in the following sections.

A. Information Management: Sender Agent

The purpose of the sender agents is to manage information by holding or delaying the messages and processing them based on the priorities. When the sender sends a message from the chat interface, the sender agent first intercepts and withholds it for further processing. When the sender sends the first message, the sender agent generates the First Notification Time (FNT) for itself and for a receiver agent to become aware of the message waiting for delivery. The generated FNT remained unchanged until all the queued messages are delivered. The process initiation for the sender agent is shown in Algorithm 1.

Algorithm 1 Framework Initiation

```
1: procedure INITIATEFRAMEWORK
2: Sender sends a message for delivery to Receiver
3: Sender Agent intercepts the message and queues it for delivery
4: Sender Agent generates First Notification Time indicating message waiting for delivery
5: end procedure
```

After Algorithm 1, the sender agent gathers contextual information to determine the priority of the message, which is influenced by two kinds of information: One piece of information is the context of the message and the other is related to the sender’s behavior.
The context of the message is in the message itself. The agent checks for a keyword such as “Urgent” or “Important” in the single message or different lines of messages and judges the urgency. If the message contains either of these keywords then a flag $\delta$ is set to True. Otherwise, it is set to False. We assign a binary value to represent the presence of the keyword.

The information related to the sender’s behavior are: Message Frequency, Visiting Frequency, and Continuous Focus. The message frequency is the number of messages sent by the sender after FNT is generated. It is represented by ‘$\alpha$’ and holds the count of messages. The visiting frequency represented by $\beta$ is the number of times the sender opens the same chat application after FNT. We give thresholds ‘C’ and ‘D’ for ‘$\alpha$’ and ‘$\beta$’ respectively, and use them for judging the message priority. The continuous focus tracks whether the sender has continuously opened the chat interface as an active application without switching to other applications. It is represented by ‘$\gamma$’ and holds True or False. If the sender is anticipating a reply to an urgent message, then we can consider that the chat interface will remain as an active application until replied.

Along with the contextual information, it calculates the message priority for that particular sender. The message priorities are derived from the combination of the two kinds of information mentioned above. They are classified into 3 levels and collectively represented as Message Priority (MP): High (x), Normal (y), and Not Specified (z), where $x, y, z$ are numerical values and $x > y > z$. The values assigned to the different message priority levels determine the Information Management Point (IMP) for the respective senders. The rules to decide values of ‘MP’ and ‘IMP’ from $\alpha$, $\beta$, $\gamma$, and $\delta$ are shown in Table I as an example. They could be modified to adjust the actual work environments of the users contributing to information management. Therefore, it requires co-operating with the receiver agent for the appropriate time to deliver the messages depending on their urgency. This cooperation results in the process of calculating the importance of the message, i.e., whether the information contained in the message is worth to deliver at the particular moment. The process of information management can be perceived in Algorithm 2.

### Algorithm 2 Calculate IMP

1. procedure CALCULATEIMP(sender)
2. $\alpha \leftarrow$ get sender’s message frequency
3. $\beta \leftarrow$ get sender’s visiting frequency
4. $\gamma \leftarrow$ check if sender is continuously focusing on chat window
5. $\delta \leftarrow$ check if keyword is present
6. for each records in Table I do
7. if ($\alpha$ satisfies the condition set for TableI.$\alpha$) and ($\beta$ satisfies the condition set for TableI.$\beta$) and ($\gamma$ matches TableI.$\gamma$) and ($\delta$ matches TableI.$\delta$) then
8. MP = TableI.MP
9. IMP = TableI.MP
10. end if
11. end for
12. return IMP
13. end procedure

The contextual and behavioral information help the sender agent to determine and set the Message Priority (MP) automatically liberating the senders from setting message priority manually. This approach is a better idea because if senders are allowed to set the message priority themselves, they might want to deliver the message at the earliest and sets their MP as High without apprehending about the receiver’s convenience.

### B. Interruption Management: Receiver Agent

At the receiver’s end, this framework manages and reduces interruptions to the receiver caused by message delivery via a chat interface. A receiver agent is introduced here to reschedules the incoming messages based on two different kinds of factors until the receiver’s favorable time is reached. These factors are Sender-Receiver Relation (SRR) and message Waiting Duration (WD).

**SRR** classifies the senders based on their relation to the receiver and determines how important is that sender’s message to the receiver. Initially, all the senders in the receiver chat list will have SRR as ‘Normal (c)’ until the receiver manually changes these to either ‘Most Important (a)’ or ‘Important (b)’, where $a, b, c$ are numerical values and $a > b > c$. Senders who have High SRR with receiver gets high priority to deliver the message first.

The next factor, WD, is the time duration the sender is waiting to deliver the messages. For each sender, this time elapses after first notification FNT is received by the receiver. To cater for situations where the sender’s message is waiting for an extended amount of time, we introduce a concept of Minimum Wait Duration (MWD). In the chat application, $MWD$ is an option to set the minimum wait time duration after which the message needs to be delivered sooner. When the message waiting time WD passes MWD, then the Wait (W) becomes High (h) otherwise, it will be Low (l); $h$ and $l$ being numerical values where $h > l$.

Then, the receiver agent sums up SRR and W to ensure that the message from that sender is worth to deliver without

---

**TABLE I**

<table>
<thead>
<tr>
<th>S.N</th>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$\gamma$</th>
<th>$\delta$</th>
<th>MP</th>
<th>IMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$&gt; C$</td>
<td>$&gt; D$</td>
<td>True</td>
<td>True</td>
<td>High</td>
<td>$x$</td>
</tr>
<tr>
<td>2</td>
<td>$&gt; C$</td>
<td>$&gt; D$</td>
<td>False</td>
<td>True</td>
<td>High</td>
<td>$x$</td>
</tr>
<tr>
<td>3</td>
<td>$&gt; C$</td>
<td>$&gt; D$</td>
<td>True</td>
<td>False</td>
<td>Normal</td>
<td>$z$</td>
</tr>
<tr>
<td>4</td>
<td>$\leq C$</td>
<td>$&gt; D$</td>
<td>False</td>
<td>False</td>
<td>Normal</td>
<td>$y$</td>
</tr>
<tr>
<td>5</td>
<td>$\leq C$</td>
<td>$\leq D$</td>
<td>True</td>
<td>True</td>
<td>High</td>
<td>$x$</td>
</tr>
<tr>
<td>6</td>
<td>$\leq C$</td>
<td>$\leq D$</td>
<td>True</td>
<td>False</td>
<td>Normal</td>
<td>$y$</td>
</tr>
<tr>
<td>7</td>
<td>$\leq C$</td>
<td>$\leq D$</td>
<td>False</td>
<td>True</td>
<td>Normal</td>
<td>$y$</td>
</tr>
<tr>
<td>8</td>
<td>$\leq C$</td>
<td>$\leq D$</td>
<td>False</td>
<td>False</td>
<td>Not Specified</td>
<td>$z$</td>
</tr>
<tr>
<td>9</td>
<td>$\leq C$</td>
<td>$\leq D$</td>
<td>True</td>
<td>True</td>
<td>High</td>
<td>$x$</td>
</tr>
<tr>
<td>10</td>
<td>$\leq C$</td>
<td>$\leq D$</td>
<td>True</td>
<td>False</td>
<td>High</td>
<td>$x$</td>
</tr>
<tr>
<td>11</td>
<td>$\leq C$</td>
<td>$\leq D$</td>
<td>False</td>
<td>True</td>
<td>Normal</td>
<td>$y$</td>
</tr>
<tr>
<td>12</td>
<td>$\leq C$</td>
<td>$\leq D$</td>
<td>False</td>
<td>False</td>
<td>Normal</td>
<td>$y$</td>
</tr>
<tr>
<td>13</td>
<td>$\leq C$</td>
<td>$\leq D$</td>
<td>True</td>
<td>True</td>
<td>High</td>
<td>$x$</td>
</tr>
<tr>
<td>14</td>
<td>$\leq C$</td>
<td>$\leq D$</td>
<td>True</td>
<td>False</td>
<td>Normal</td>
<td>$y$</td>
</tr>
<tr>
<td>15</td>
<td>$\leq C$</td>
<td>$\leq D$</td>
<td>False</td>
<td>True</td>
<td>Normal</td>
<td>$y$</td>
</tr>
<tr>
<td>16</td>
<td>$\leq C$</td>
<td>$\leq D$</td>
<td>False</td>
<td>False</td>
<td>Normal</td>
<td>$y$</td>
</tr>
</tbody>
</table>
interrupting the receiver. The value obtained after summation of SRR and W is called Interruption Management Point (IntMP). The reason behind calculating the IntMP value as the sum of SRR and W is described as follows. When the message is sent from an important person (SRR = Most Important), then the receiver would prefer to attend the message instead of holding it, so it would not be considered as an interruption. Even an interruptive message waiting for some time needs to be delivered with managed delay as this framework manages both interruption and information. Interruption management or the receiver agent’s process can be visualized in Algorithm 3.

### Algorithm 3 Calculate IntMP

1: procedure CALCULATEINTMP(receiver)
2:  get Sender Receiver Relation
3:  if Sender Receiver Relation = Most Important then
4:    SRR ← a
5:  end if
6:  if Sender Receiver Relation = Important then
7:    SRR ← b
8:  end if
9:  if Sender Receiver Relation = Normal then
10:   SRR ← c
11: end if
12: WD ← waiting time elapsed duration since FNT
13: MWD ← minimum wait duration
14: if WD > MWD then W ← h
15: elseW ← 1
16: end if
17: IntMP ← SRR + W
18: return IntMP
19: end procedure

Apart from above tasks, the receiver agent also keeps track of receiver’s contextual information such as the time of application switching by the receiver, setting regular time intervals, and the time when the receiver take an intermission. Intentional switching of work is application switching, whereas intermission is a pause or a break from regular work. The reason behind considering this contextual information is because these are the alternative ways for notification presentation to manage interruption [2].

### C. Receiver Oriented Interruption-Information Management

This framework conceptualizes a chat interface consisting of both the sender agents and receiver agent. Multiple sender agents and a receiver agent process the messages, message contexts as well as other factors to confer a message delivery time so that it would be less-interruptive for the receiver. When a sender agent generates the first notification FNT, the receiver agent acknowledges the notification but denies immediate message delivery to the receiver. At the same time, both the sender agent and the receiver agent start gathering the contextual information and calculate MP, SRR, and W to obtain interruption management point IntMP and information management point IMP at their respective ends. Then, the sender agent cooperates with the receiver agent based on IMP and IntMP. In the case of multiple senders, the decision on whose message to deliver first depends on the summation of IntMP and IMP. This value indicates the importance of the message from the perspective of both interruption and information management. So, the sender-receiver pair with the highest sum of IntMP and IMP gets the first chance to deliver the message.

Sender-Receiver Relation (SRR), Wait (W) and Message Priority (MP) are the bargaining factors for both the sender agent and the receiver agent. Based on the different combination of these traits, the message is scheduled to deliver either Immediately (Imm), during Application Switching (AS), Regular Interval (RI) or Self Intermission (SI), as shown in Table II. The rules in the table can be modified to adjust the actual work environments. As mentioned earlier, these four conditions are the available alternative ways of notification presentation. Regarding interruption management, Immediate, Application Switching, Regular Interval and Self Intermission exhibit the better interruption management in an increasing order [2].

Finally, when the most favorable Message Delivery Condition occurs at receiver agent’s end, all the queued messages get delivered to the specific sender. This process marks the end of the round of message delivery. All of the queued messages will be delivered when the receiver becomes in intermission mode. The process of interruption-information management can be visualized in Algorithm 4.

### Algorithm 4 Interruption Information Framework

1: procedure INTERRUPTIONINFORMATIONMANAGEMENT
2:  while chat application is running do
3:    if Notification = True then
4:      for each sender do
5:        α ← get sender’s message frequency
6:        if α ≥ 1 then
7:          IMP = IMP + 1
8:          IntMP = CalculateIntMP(sender)
9:        end if
10:      end for
11:    end if
12:    for each sender do
13:      Get Sender with Max(TotalPoints)
14:      Calculate MessageDeliveryCondition(SRR, W, MP) for sender having Max(TotalPoints)
15:      Deliver all the queued messages that satisfies MDC
16:    end for
17:    clear FNT
18:  end while
19: return
20: end procedure

We empirically test this framework to assure that it satisfies the designed purpose and resolves the current issues of interruption and information management in chat interface. The primary objective of this study is to maximize productivity by minimizing interruptions.

### IV. EXPERIMENTAL PROCEDURE

This experiment is focused towards users’ working on specific tasks who employ chat application for work or non-work related communication. The objective of this experiment is to gauge users’ productivity and satisfaction level on two different chat strategies. The same experiment is conducted twice in random order with each user: i) with conventional
TABLE II
Determining Message Delivery Condition

<table>
<thead>
<tr>
<th>S.N</th>
<th>SRR</th>
<th>W</th>
<th>IMP</th>
<th>MDC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>l</td>
<td>x or y</td>
<td>Imm</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>l</td>
<td>z</td>
<td>AS</td>
</tr>
<tr>
<td>3</td>
<td>a</td>
<td>h</td>
<td>x</td>
<td>Imm</td>
</tr>
<tr>
<td>4</td>
<td>a</td>
<td>h</td>
<td>y or z</td>
<td>Imm</td>
</tr>
<tr>
<td>5</td>
<td>b</td>
<td>l</td>
<td>x</td>
<td>AS</td>
</tr>
<tr>
<td>6</td>
<td>b</td>
<td>l</td>
<td>y or z</td>
<td>AS</td>
</tr>
<tr>
<td>7</td>
<td>b</td>
<td>h</td>
<td>x</td>
<td>Imm</td>
</tr>
<tr>
<td>8</td>
<td>b</td>
<td>h</td>
<td>y or z</td>
<td>AS</td>
</tr>
<tr>
<td>9</td>
<td>c</td>
<td>l</td>
<td>x</td>
<td>RI</td>
</tr>
<tr>
<td>10</td>
<td>c</td>
<td>l</td>
<td>y or z</td>
<td>SI</td>
</tr>
<tr>
<td>11</td>
<td>c</td>
<td>h</td>
<td>x</td>
<td>RI</td>
</tr>
<tr>
<td>12</td>
<td>c</td>
<td>h</td>
<td>y or z</td>
<td>RI</td>
</tr>
</tbody>
</table>

Algorithm 5 Message Delivery Condition

1: procedure MESSAGE_DELIVERY_CONDITION(SRR, W, MP)
2: for each record in Table II do
3:   if sender’s SRR = TableII.SRR AND sender’s W = TableII.W
4:     MDC = TableII.MDC
5:   end if
6: end for
7: return MDC
8: end procedure

This experiment envisions 2 different tasks: 1) Typing tutor and 2) Temperature check. The Typing Tutor task requires participants to go to an online typing tutor website and type the words as they appear on the screen. This experiment is a time-limited and needs high concentration to complete the test on time while maintaining the accuracy. The participants are asked to perform two sets of typing tests with the first set being mandatory, whereas the next being optional. Each set consists of several small typing tests. The second task is a temperature checking task. The experimenter asks the participants to find out the current temperature of a given city using the internet and reply back. The task is intended to cause interruption. The experimenter occasionally asks some general questions such as academic background and hobbies as a part of interruption. The participants are directed to attend the chat messages as soon as possible. The interruption occurs in every 2 minutes.

V. EXPERIMENTAL RESULT

The productivity analysis in Table III clearly indicates that participants work more productively using interruption-information management chat than conventional chat as they completed comparatively many tests with higher accuracy and speed. Although, the time spent on non-task related chat was relatively similar or higher using interruption-information management chat most of the participants saved a good amount of time after completing the first set of typing test which was mandatory. The productivity analysis showed the positive result which is due to the minimized interruption frequency in interruption-information management chat application as shown in Table IV.

The time spent analysis in Figure 1 distinctly shows that task completion time is less in Interruption-Information Management chat as compared to the conventional one. After completing the first set of typing task, the time saved is used to work on the second set of typing tutor in interruption-information management chat, whereas in conventional chat, the participants got very short time to work on the second set. Hence, this empirical test is evidence that Interruption-Information Management chat increases users’ overall productivity and decreases interruption frequency.

VI. DISCUSSION

Our approach tries to solve the problem of users receiving myriad of annoying messages consistently which deviates them from concentrating on their task. When the interruptive messages are subdued and prioritized, the users spend much time on their task leading to lesser task fragmentation. Hence, the users become more productive using the new approach to the chat system and enrich the quality of work as they can concentrate for a longer span of time.

At the beginning of the experiment, all of the participants were asked to set the SRR according to task priority, but none
TABLE III
PRODUCTIVITY ANALYSIS

<table>
<thead>
<tr>
<th>Analysis Parameters</th>
<th>Participant 1</th>
<th>Participant 2</th>
<th>Participant 3</th>
<th>Participant 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Spent on Task (mins)</td>
<td>15:36</td>
<td>14:01</td>
<td>14:52</td>
<td>12:56</td>
</tr>
<tr>
<td>Time Spent on Chat/ non-Task (mins)</td>
<td>4:24</td>
<td>4:25</td>
<td>4:33</td>
<td>5:31</td>
</tr>
<tr>
<td>Time Saved after completing Task (mins)</td>
<td>0:00</td>
<td>1:34</td>
<td>0:35</td>
<td>1:33</td>
</tr>
<tr>
<td>Average Accuracy (%)</td>
<td>95</td>
<td>96</td>
<td>93</td>
<td>95</td>
</tr>
<tr>
<td>Average Speed (WPM)</td>
<td>29</td>
<td>33</td>
<td>28</td>
<td>35</td>
</tr>
<tr>
<td>No. of Test Completed</td>
<td>13</td>
<td>14</td>
<td>13</td>
<td>14</td>
</tr>
</tbody>
</table>

Fig. 1. Productivity based on time spent on completing the task
Con : Conventional Chat and Int : Interruption-Information Management Chat

TABLE IV
INTERRUPTION FREQUENCY

<table>
<thead>
<tr>
<th>Chat Application</th>
<th>Interruption Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1-P4 are equivalent to Participant 1- Participant 4</td>
<td></td>
</tr>
<tr>
<td>Conventional</td>
<td>P1</td>
</tr>
<tr>
<td>Interruption-Information Management</td>
<td>5</td>
</tr>
</tbody>
</table>

VII. CONCLUSION AND FUTURE WORK

This paper takes the first step toward interruption-information management in human-computer interaction. We develop a framework where the message delivery time is scheduled based on users’ preferences and changing behaviors. This is distinct from available research as this is based on automated agents’ cooperating to deliver messages at the most favorable time. In this paper, the study is limited to the verification of the designed approach and productivity analysis. Further, we will perform preference-based analysis to study the relation between message delivery condition and the message. In the future, we can also incorporate the chat history data to assess the sender-receiver relation automatically without user’s input.

REFERENCES