Wedjat: A Mobile Phone Based Medication Reminder and Monitor

Mei-Ying Wang, John K. Zao
Computer Science Department
National Chiao Tung University
Hsinchu, Taiwan, Republic of China
chesnut.cs96g@nctu.edu.tw, jkzao@cs.nctu.edu.tw

P.H. Tsai, J.W.S. Liu
Institute of Information Science
Academia Sinica
Taipei, Taiwan, Republic of China
peipei@iis.sinica.edu.tw, janeliu@iis.sinics.edu.tw

Abstract— **Out-patient medication administration has been identified as the most error-prone procedure in modern healthcare. Under or over doses due to erratic in-takes, drug-drug or drug-food interactions caused by un-reconciled prescriptions and the absence of in-take enforcement and monitoring mechanisms have caused medication errors to become the common cases of all medical errors. Most medication administration errors were made when patients bought prescribed and over-the-counter medicines from several drug stores and use them at home without little or no guidance. Elderly or chronically ill patients are particularly susceptible to these mistakes.**

In this paper, we introduce *Wedjat*, a smart phone application designed to help patients avoiding these mistakes. *Wedjat* can remind its users to take the correct medicines on time and record the in-take schedules for later review by healthcare professionals. *Wedjat* has two distinguished features: (1) it can alert the patients about potential drug-drug/drug-food interactions and plan a proper in-take schedule to avoid these interactions; (2) it can revise the in-take schedule automatically when a dose was missed. In both cases, the software always tries to produce the simplest schedule with least number of in-takes. *Wedjat* is equipped with user friendly interfaces to help its users to recognize the proper medicines and obtain the correct instructions of taking these drugs. It can maintain the medicine in-take records on board, synchronize them with a database on a host machine or upload them onto a Personal Heath Record (PHR) system. A proof-of-concept prototype of *Wedjat* has been implemented on Window Mobile platform and will be migrated onto Android for Google Phones. This paper introduces the system concept and design principles of *Wedjat* with emphasis on its medication scheduling algorithms and the modular implementation of mobile computing application.

***Keywords*— telemonitoring, medication error prevention, mobile computing, real-time scheduling.**

I. INTRODUCTION

According to a landmark study on medical errors conducted by the US Institute of Medicine in 1999 [1], *medication errors (ME)* and *adverse drug reactions (ADR)* are the most common cases among all medical errors. These *adverse drug events (ADE)* incurred significant tolls in terms of patient fatality, financial costs (including additional medical expenses, income and productivity losses), and damages to the reputation and morale of healthcare professionals. Most of these errors are nonetheless preventable [2]. One study found 530,000 preventable ADEs among Medicare out-patients each year.

Although errors can occur in every step of medication process during medicine procurement, prescription, dispensing and administration, they happen most frequently in the *prescription* and *administration* stages. In the past decade, increasingly wide-spread use of computerized physician order entry (CPOE) [3], clinical decision support systems (CDSS) [4] and electronic medical records (EMR) [5] along with better procedures to dispense medicine has helped to eliminate a large proportion (up to 80%) of prescription errors, which account for half of all medication errors. In comparison, little progress has been made in the prevention of administration errors, which are caused by improper and failed use of prescribed medicine. Consequently, medication administration errors have become the prevalent cause of ADEs. They accounted for 25%–40% of all medication errors and were the main reason for admission of elderly into nursing homes [6].

Out-patient medication administration has been identified as the most error prone procedure amidst the entire medication process. Most of these errors were made when patients bought prescribed and over-the-counter (OTC) medicines from different drug stores and use them at home without little or no guidance. Common causes of these errors include: (1) *irregular medicine in-takes* due to the patient's busy or erratic lifestyles, (2) *complicated in-take schedules* due to many medicines and doses taken by the patient, (3) *adverse drug reactions* caused by un-reconciled prescriptions obtained from different sources, (4) *lack of knowledge* about proper use of medicines, (5) *lack of consultation* with healthcare providers when confusion arises and (6) *lack of monitoring mechanisms* to keep track of patient's medicine in-take. Recently, telemedicine, especially *telemonitoring* techniques, has been investigated as a cost-effective approach to control quality of care (QoC) in out-patient medication administration [7,8,9, 10]. By sending in-take reminders to the patient, producing the proper medicine from a medicine dispenser and recording patient's in-take schedules, Health Maintenance Organizations (HMO) hope to reducing cost of service while improving quality of care. Communication between HMOs and patients is established through wired or wireless Internet connections. Although these efforts represent progress in the right direction, the medicine dispensers thus made are bulky, expensive and prone to dispensing errors. A handy alternative solution can be provided by installing a *medication reminder and monitor* on a smart mobile phone and then using it along with a traditional mechanical "pill box". Such a solution will be cheaper (excluding the incurred cost of the smart phone) and may result in deeper penetration into the consumer market.

In this paper, we introduce *Wedjat*, a smart phone application designed to help patients to avoid medicine administration errors. The application was named after the "Eye of Egyptian God Horus" [cf. insert] from which derived the prescription symbol R_x. *Wedjat* can perform three primary functions:

1. *Issue medicine in-take reminders — Wedjat* will issue an alert approximately 5 – 15 minutes (preset by user) before the scheduled time to take certain medicine(s). The alert will be issued repetitively until it is cancelled by the user. Scheduling of in-take alerts is performed by a real-time process/resource scheduling algorithm [Section IV] that can satisfy time constraints according to medicine in-take directions and drug-drug/drug-food interactions. This function is integrated with the calendar and planner applications installed on most smart phones.
2. *Provide medicine identification and in-take directions — Wedjat* has a built-in database containing crucial information about the medicines (including their photo images, in-take directions and precautions) and the healthcare providers (including physicians, pharmacists and HMOs) relevant to its user. All these data can be retrieved with the touch of a button while *Wedjat* is in use.
3. *Maintain medicine in-take records — Wedjat* will record the time at which its user cancels an in-take alert and regard that at the time that specific medicine(s) was taken. These medicine in-take records can be stored on board, synchronize with the database on a host machine and/or uploaded onto a Personal Heath Record (PHR) system.

Wedjat has two distinguished features: (1) it can alert the patients about potential drug-drug/drug-food interactions and plan an in-take schedule to avoid these interactions; (2) it can revise the in-take schedule automatically when a dose was missed. In both cases, the software always tries to produce the simplest schedule with least number of in-takes. A proof-of-concept prototype of *Wedjat* has been implemented on the Window Mobile platform and will be migrated onto Android for Google Phones.

The rest of this paper presents the design concepts of *Wedjat* with emphasis on its medication scheduling algorithms and modular implementation. An overview of the *Wedjat* system is given in Section II. It is followed by discussions of the formal specification of the time constraints [Section III] and the two algorithms used in medicine in-take scheduling [Section IV]. Actual implementation of the mobile computing application is discussed in Section V. Related work is compared in Section before accomplishments and future work are summarized in a brief conclusion.

II. System Overview

A. System Architecture

Figure 1 shows the system architecture of *Wedjat*. The system uses an electronic medical record (EMR) or an electronic personal health record (EPHR) system as the consolidator of patients' medication prescriptions and in-take records. In our proof-of-concept prototype, an open-source EPHR system *Indivo* [11] was chosen to play that role.

Medication prescriptions are loaded into the secure persistent database of Indivo from medical clinics, hospitals and pharmacies. The prescriptions are checked for potential drug-drug and drug-food interactions and converted into an XML-formatted medication scheduling specification (MSS) [Section III.A].

Patient's MSS files are sent to his/her mobile phone via RSS feeds [12]. The *Wedjat* mobile application running on patient's mobile phone then computes the medication schedules based on the MSS data and inserts alarm triggering events into the mobile phone calendar.

Wedjat also records patient's medicine in-take time in a medication in-take record (MIR) file when he/she presses the reset button to cancel the alarm. The MIR files may be uploaded onto the Indivo server at prescheduled time or by patient's command. Physicians and healthcare providers may review the MIS files and change patient's prescriptions by uploading revised ones onto Indivo and pushing the MSS to patient's mobile phones. The exchanges of MSS and MIR files thus provide the healthcare providers with an on-line mechanism for monitoring the medication administration to the out-patients and adjust their medication doses without additional clinical visits.

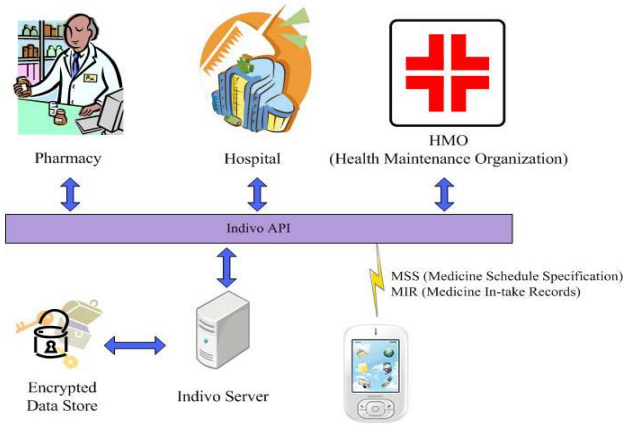

Figure 1 Wedjat System Architecture

B. Operation Scenarios

For every in-take, *Wedjat* inserts an alarm triggering event into the mobile phone calendar, which will then cause an alarm to be set off at a preset time before the proper in-take time is due. If the patient takes the medicines and reset the alarm, *Wedjat* will store the in-take time in a medication in-take record (MIR) file. If the patient forgot to take his/her medicines, *Wedjat* will remind him/her again until the in-take period expires. In that case, it will compute a new medication schedule and remind the patient when the next in-take time is due. If *Wedjat* could not produce a viable schedule that

satisfies the MSS, it will issue a warning and suggest the patient to contact his/her healthcare provider.

III. Medicine Prescription and Scheduling Specification

A. Medication Scheduling Specification

Table 1 shows the medication scheduling specification (MSS) of a specific medicine *M*. The MSS can be divided into three parts containing the prescription parameters, the dosage parameters and the interaction parameters.

Prescription Parameters (PP)	
Medicine Identifier	*M*
Medicine Dose	*g*
Medicine Form	Capsule/Tablet/…
Medicine Amount	*n*
Therapy Duration	*T*
Dosage Parameters (DP)	
Nominal Min. & Max. Dose	`[dmin, dmax]`
Nominal Min. & Max. Separations	`[nsmin, nsmax]`
Max. In-take Amount *B* over Interval *R*	`(B, R)`
Min. Intake Amount *L* over Interval *P*	`(L, P)`
Absolute Min. & Max. Dose	`[Dmin, Dmax]`
Absolute Min. & Max. Separations	`[asmin, asmax]`
Interaction Parameters (IP) <List>	
Interferer Identifier	*N*
Min. Separation from M to N	minToInterferer
Min. Separation from N to M	minFrInterferer

Table 1: Sample Medication Scheduling Specification

1) Prescription Parameters (PP)

These parameters contain essential information of a medicine including its *name M*, its *dosage size g* and the *amount n* to be taken each time (in terms of multiples of dosage size) and its *therapy duration T during which* the patient should take the medicine. A picture of the medicine is also included to help patients recognizing the medicine.

2) Dosage Parameters (DP)

These parameters specify the constraints imposed on dose size and separation (i.e., the duration between two consecutive doses) for scheduling the medicine. Take Advil for example, its direction is "to take one capsule every 4 to 6 hours. If pain or fever continues, two capsules may be used." In this example, the nominal dose size and separation of Advil are `[1,2]` and `[4,6]`, respectively.

Supply Rate `(B,R)` implies that the maximum dosage *taken* in any time interval of length *R cannot exceed the amount B*. For example, the supply rate of Advil is `(6,24)` because its direction also says "Do not exceed 6 capsules in 24 hours."

Demand Rate `(L,P)` implies that the minimum dosage taken in any time interval of length `P` *cannot be below the amount* `L`.

Dosage parameters may also include hard constraints such as *absolute dose sizes* `[Dmin, Dmax]` and *absolute separations* `[asmin, asmax]`. By making these absolute ranges wider than the nominal ranges, we allow some flexibility in worst-case scheduling.

3) Interaction Parameters (IP)

We refer to a medicine or food that interacts with *M* as an *interferer N* of *M*. The IP section of *M* contains an entry for each of its interferers.

The IP entry of each interferer *N specifies* two separation constraints: `minToInterferer`, which indicates the *minimum separation from the medicine M to the interferer N*, and `minFrInterferer`, which indicates the *minimum separation from the interferer N to the medicine M*. Take Fosamax as an example. It must be taken on empty stomach; also the patient should not take anything within 30 minutes after taking the medicine. Thus, the minimum separation parameters to and from any interferer of Fosamax are 0.5 hour and 6 hours, respectively.

IV. Medication Scheduling Algorithms

Medication scheduling is the core function of *Wedjat*. It is performed by two scheduling algorithms known as *one-medicine-at-a-time (OMAT)* and *one-dose-at-a-time (ODAT) scheduling*, which are devised according to real time scheduling theory. This section provides an overview of these two algorithms along with their resource models, resource allocation rules and scheduling priority schemes.

A. Scheduling Models

1) Resource Model

Both OMAT and ODAT use a resource model [13,14] that employs a *virtual processor PM* to keep track of patient's availability to take a medicine *M* and a *virtual resource RM* to account for potential interactions between that medicine and the others. When computing a schedule, the algorithms treat each dose of each medicine *M* as a *job* running on processor *PM* and the sequence of doses of *M* as a *task*. A job starts when the dose it represents is taken by the user. A *schedule* for a medicine *M* is a list of time at which jobs/doses of task/ medicine *M* start.

2) Resource Allocation Rules

The scheduling algorithms maintains correct separations between consecutive doses of a medicine by scheduling each dose as a non-preemptive job running on *PM* for a minimum duration of `dsmin`. In addition, the absolute maximum separation between successive doses is enforced by imposing a relative deadline for each dose/job.

To avoid drug-drug and drug-food interactions, the algorithms use a virtual resource *RM* to maintain minimum separations

```
INPUT:
  ListMedicationScheduleSpec MSS[number_medicines]
  PrioritySchemes PS
OUTPUT:
  Boolean feasibleSchedule
  List MedicineIntakeSchedule MIS[number_medicines]
PROCEDURE:
  FOR every MSS[i]{
    CREATE JobModel, ResourceModel
    CREATE integer latestStartTime }
  ASSIGN priority to every MSS[i]
    based on PrioritySchemes PS
  SET feasibleSchedule to TRUE
  FOR every MSS[i] according to descending priority{
    SET latestStartTime of MSS[i] to 0
    SET ResourceModel.feasible of MSS[i] to TRUE
    WHILE ResourceModel.feasible of MSS[i] is TRUE {
      FOR i from JobModel.release_time to
           JobModel.deadline of MSS[i] {
        IF ResourceModel.processor[i]==0 and
           ResourceModel.resource[i]==0 {
          SET Tavail to I THEN BREAK }}
      IF Tavail is found
        IF Tavail>=MSS[i].TheraphyDuration THEN BREAK
        APPEND Tavail to ResourseModel.Schedule
        SET latestStartTime of MSS[i] to Tavail
        SET JobModel.release_time to (Tavail+ nsmin)
        SET JobModel.deadline to
            (JobModel.release_time + asmax)
        FOR k from Tavail to (Tavail + nsmin)
          SET ResourceModel.process[k] of MSS[i] to 1
        FOR every interferer MSS[i].IP[j]{
          FOR k from {
          (Tavail - MSS[i].IP[j].minFrInterferer) to
          (Tavail + MSS[i].IP[j].minToInterferer)
            SET ResourceModel.resource[k] of
                MSS[MSS[i].IP[j].med_id] to 1 }}
      ELSE
        SET ResourceModel.feasible of MSS[i] to FALSE
        SET feasibleSchedule to FALSE }
    IF feasibleSchedule is TRUE {
      SET MIS[i] to ResourceModel.Schedule of MSS[i]}
  RETURN feasibleSchedule and MIS[]
END PROCEDURE
```

Table2: OMAT Pseudo-code

```
INPUT:
  List MedicationScheduleSpec MSS[number_medicines]
  PrioritySchemes PS
OUTPUT:
  Boolean feasibleSchedule
  List MedicineIntakeSchedule MIS[number_medicines]
PROCEDURE:
  FOR every MSS[i]
    CREATE JobModel, ResourceModel
  SET numberMedicines to the length of MSS[] array
  CREATE integer array
    ReleaseTimeCurrentJobs[numberMedicines]
  SET ScheduleComplete to 0
  SET feasibleSchedule to TRUE
  WHILE (feasibleSchedule is TRUE) and
      (scheduleComplete < numberMedicines){
    FIND MSS[i] with earliest time in
      ReleaseTimeCurrentJobs[] and highest priority
    FOR k from JobModel.release_time
         to JobModel.deadline of MSS[i]{
      IF ResourceModel.processor[k]==0
        and ResourceModel.resource[k]==0{
        SET Tavail to k  THEN BREAK }}
    IF Tavail is found
      IF Tavail >= MSS[i].TheraphyDuration {
        INCREMENT scheduleComplete THEN BREAK }
      APPEND Tavail to ResourseModel.Schedule
      APPEND ResourceModel.Schedule to MIS[i]
      SET JobModel.release_time to (Tavail+nsmin)
      SET JobModel.deadline to
          (JobModel.release_time + asmax)
      SET ReleaseTimeCurrentJobs[i] to(Tavail+nsmin)
      FOR k from Tavail to (Tavail + nsmin)
        SET ResourceModel.process[k] of MSS[i] to 1
      FOR every interferer MSS[i].IP[j] {
        FOR k from
        (Tavail - MSS[i].IP[j].minFrInterferer) to
        (Tavail + MSS[i].IP[j].minToInterferer) {
          SET ResourceModel.resource[k] of
              MSS[MSS[i].IP[j].med_id] to 1 }}
    ELSE
      SET feasibleSchedule to FALSE }
  RETURN feasibleSchedule and MIS[]
END PROCEDURE
```

Table 3: ODAT pseudo-code

between a medicine and its interferers. The algorithms allocate the resource *RM* to each job/dose of an interferer *N*

for the duration of `minFrInterferer` when it schedules a job/dose on *PM*. IN other words, each job of the interferer *blocks* a job/ dose of the interfering medicine *M* during `minFrInterferer`.

3) Priority Schemes

Both OMAT and ODAT operate based on priorities. Plausible priority schemes include *Most Interferers First* (*MIF*) and *Shortest Separation Difference First* (*SSDF*) schemes. They give higher priorities to the tasks corresponding to medicines with more interferers or larger differences between maximum and minimum nominal separations, respectively. The best priority schemes is the *Most Victimized First* (*MVF*) scheme, which assigns priorities to the tasks based on their worst case blocking times — the longer the worst case blocking time, the higher the priority of the task/medicine.

B. Scheduling Algorithms

Table2 and Table **3** list the pseudo-codes of basic OMAT and ODAT algorithms. In those codes, the Boolean variable `feasible` indicates whether the algorithms succeeded in finding a feasible schedule when they terminate. If succeeded, the array `feasibleSchedule` point to the schedules of the medicines.

1) One-Medicine-at-A-Time (OMAT)

The OMAT algorithm schedules one medicine at a time according to a descending order of the importance of the medicine towards the patient. Every dose of the medicine to be taken by the patient during its therapy duration is scheduled before the next medicine will be considered. Interactions between the medicine and all scheduled medicines must be avoided by satisfying the minimum separation to/from the interferers.

OMAT performs better in compliance with the constraints specified by the dosage parameters (DP) and interaction parameters (IP) in the MSS. However, the schedule it

produces is inflexible. When a dose is missed, the entire schedule must be refreshed by running the algorithms once again. Thus, it should be used by users who tend to take their medicines on time.

Basic OMAT algorithms tend to schedule doses of higher priority medicines close together, leaving little or no time for doses of lower priority medicines. A simple enhancement is to schedule doses of medicines as close as to their respective deadlines as possible. We call algorithms with this enhancement *advanced OMAT* algorithms.

2) One-Dose-at-A-Time (ODAT)

The ODAT algorithm schedules the individual doses of the medicines one at a time according to dynamic importance of the doses towards the patient. ODAT select the most important dose at present to schedule. When user takes the dose, the scheduler starts plan next dose schedule in the same way. We have no idea about the schedule of other future doses before user really takes the dose. ODAT performs worse than OMAT on complying with constraints specified in MSS, but ODAT do better in deal with the real time behavior of user's delay and miss doses.

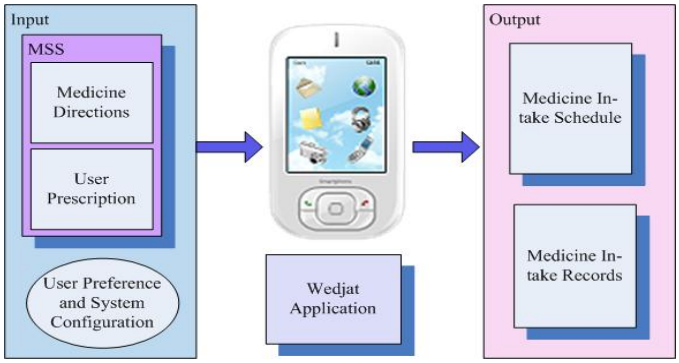

Figure 2: *Wedjat* Input/Output flow

V. PROTOTYPE IMPLEMENTATION

This section documents *Wedjat* implementation including its platform, software design and graphic user interfaces.

C. Implementation Rationale

Wedjat is built upon the .NET framework running on Windows Mobile 6.0.and is integrated with the built-in Calendar application. It accesses the Calendar application through the Calendar API included in the namespace `Microsoft.WindowsMobile.PocketOutlook`.

As shown in Figure 2, *Wedjat*'s inputs include user preferences and the Medication Scheduling Specification (MSS) of all medicines. *Wedjat* download the user's MSS from Indivo personal health record (PHR) server. In addition, user set preferences such as sleeping time and busy time by the *Wedjat* graphic user interface. The scheduler keeps away from the time slots of user's setting when possible.

Wedjat's outputs include medicine in-take schedule, medicine in-take record. After *Wedjat* create a feasible full in-take schedule, it writes the schedule to the Calendar application. If the user miss or delay some dose, the system will readjust the in-take schedule and rewrite to Calendar application. When the user clicks OK to confirm the dosage in-take, the system takes down medicine in-take records.

D. Software Design

Figure 3 shows the *Wedjat* functional modules. *Wedjat* downloads the user's MSS from Indivo. XML Parser parses the MSS XML file and then SQL Server Mobile inserts all the data into a light-weight SQL Database on mobile phone. The user sets the preference and then starts the Medication Scheduler. The Medication Scheduler schedules all medicine based on the user preference setting and MSS data query to SQL Server Mobile. If the Medication Scheduler creates feasible schedule successfully, it shows some message to graphic user interface and writes the schedule to Calendar application. The Medication Scheduler will monitor the user's behavior by tracking the event of the schedule in Calendar. When user delay or miss some doses, the Medication Scheduler creates a readjust in-take schedule and refresh the schedule in Calendar. After the user take doses, the Medication Scheduler notes down the in-take records and upload to Indivo periodically.

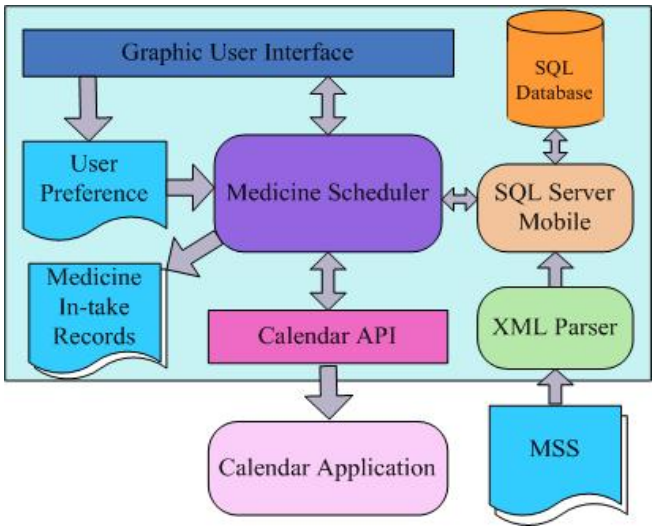

Figure 3 : *Wedjat* Functional Modules

The class diagrams of *Wedjat* are classified to two categories: Medication Schedule Specification, Medication Scheduler. As the Figure 4 shows, the data structure of MSS includes the medicine identify, Dosage Parameters, List of Interaction Parameters. It holds all constraints parameters of MSS. Medication Scheduler details in Figure 5 is the core module of *Wedjat*. It creates a `JobModel` object, a `ResourceModel` object for each medicine. The `JobModel` object holds the values of jobs parameters of each medicine M. The `deadline` field is set to the nominal minimum separation `nsmin` of M initially. The `ResourceModel` maintains two integer arrays *resource* and *processor*. The initial values of every element of the two arrays are 0 which indicates the *resource* and *processor* are free. When a job is scheduled on processor, the

processor is set to 1 for k elements in *processor* array which indicates the processor is occupied by the job for the length of execution time k. The *resource* of M is occupied by the interaction medicines or foods N of M. Only when the *processor* and M are free for the length of execution time and the *resource* is free at the start instance of job, the job can be scheduled and executed. The list schedule in `ResourceModel` is the schedule of all doses of the medicine *M* and is be set to List MIS including `med_id`, in-take time, dose size as the output of the scheduler. The Medicine Priority defines five schemes of priority including EDF, MIF, MVF, RM and SSDF. The EDF is used by ODAT only.

```
Class MSS{
  int med_id; DosageParameters dp;
  List<InteractionParameters> ip;
}
Class DosageParameters{
  int T;
  int nsmin, nsmax, asmin, asmax;
  inr dmin, dmax, Dmin, Dmax;
  int B, R, L, P;
}
Class InteractionParameters{
  int med_id; int mintoInterferer;
  int minFrInterferer;
}
```

Figure 4 : Medication Scheduling Specification Classes

```
Class MedicationScheduler{
  int feasibleSchedule;
  public bool OMAT(List<MSS> mss,
    PrioritySchemes ps,  ref List<MIS> mis);
  public bool ODAT(List<MSS> mss,
    PrioritySchemes ps,  ref List<MIS> mis);
  private void SetDoseSize(List<MSS> mss, ref
  List<MIS>  mis);
  private void SetProcessor(int latestStartTime,
    MSS mss,  ref ResourceModel rm);
  private void SetResource(int med_id,
    List<MIS> mis,  List<MSS> mss,
    ref ResourceModel rm);
}
Class ResourceModel{
  int[] resource = new int[T];
  int[] processor = new int[T];
  bool feasible = TRUE; List<int> schedule = NULL;
  int priority = 0;
}
Class JobModel{
  int release_time = 0;
  int execution_time = ns_min;
  int deadline = release_time + execution_time;
}
Class MIS{
  int med_id; int time; int dose_size;
}
Enum PrioritySchemes{RM, MVF, MIF, SSDF, EDF};
```

Figure 5 : Medication Scheduler Classes

The Medication Scheduler supports two kinds of scheduling algorithms including ODAT, OMAT. The other methods in Medication Scheduler are private methods used by the functions in Medication Scheduler only.

E. Graphic User Interfaces

The graphic user interface we design for *Wedjat* is shows as Figure 6. The first page shows three buttons for user to set preference, load a MSS Xml file, display the result schedule and a text field show the next dose user have to take. Before starting the scheduler, the user has to load the MSS file and finish preference setting. In the preference setting page, the user can set the habit of in-take medicines such as they are always take medicines on time or not and the unavailable time or special time which they don't want to take medicine. After user setting preference and loading MSS file, they can start the scheduler. The scheduler chooses an appropriate scheduling algorithm for them according their setting. If the scheduling creating in-take schedule successfully, the user can see the result in the "Display My Medication Schedule". By selecting the date, the page displays the whole day in-take schedule.

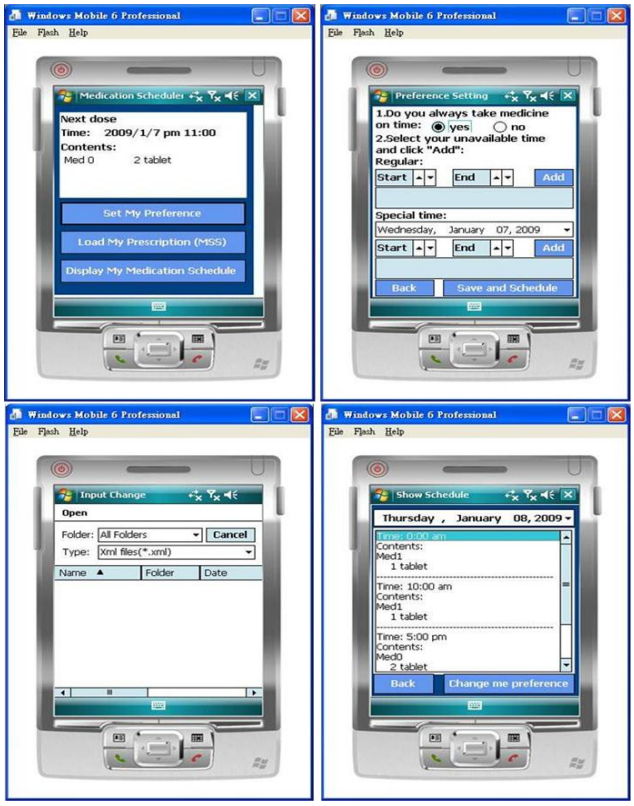

Figure 6 : Wedjat Graphic User Interfaces

F. Performance Evaluation

We used OMAT and ODAT algorithms to schedule a variety of MSS samples in several simulation experiments. We generated 1000 MSS samples for every run of simulation, which we found are sufficient to yield a 95 percent confidence. We present here a summary of the experiments and performance data of success rate. The more details of generation of MSS samples, experiment coverage and

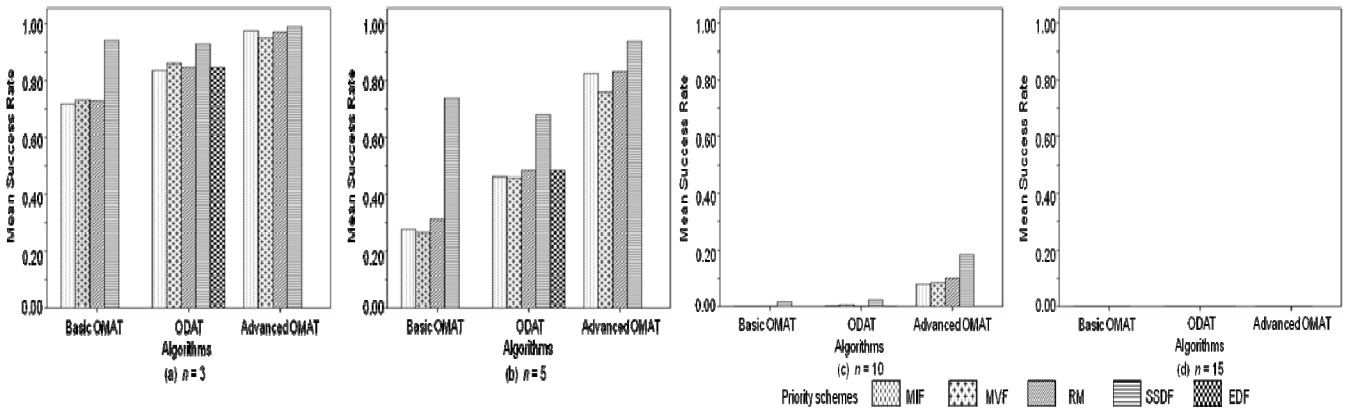

Figure 7 : Success rates of scheduling with nominal separation

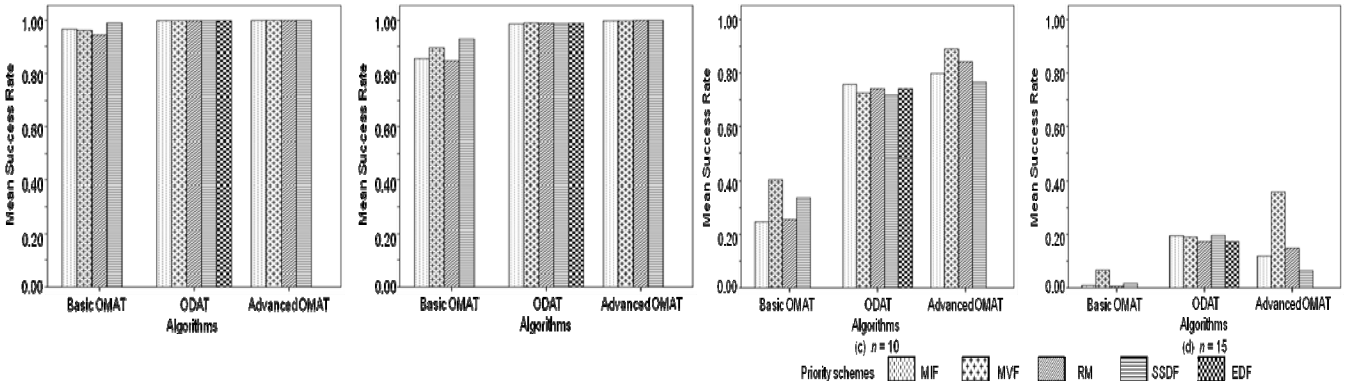

Figure 8 : Success rate of scheduling with absolute separation

additional data can be found in [15]. The success rate of an algorithm is the probability the algorithm finds a feasible schedule. Each sample MSS used in our simulation experiments is characterized by a number of parameters, including the number of medicines n, the degree medicines it interacts with other medicines. The interference probability ρ is the probability of any medicines interferers with some other medicines. The effect of interference is defined by interference severity as follow:

$$\delta(M, N) = [\texttt{minTointerferer} + \texttt{minFrinterferer}] / \max(\texttt{asmax}(M), \texttt{asmax}(N))$$

The dependence of success rate and the parameters used in simulation is tested by stepped n by 1 from 3 to 20. The parameters of ρ and δ is selected from the even distribution over the range [0.1, 1] independently. The success rate of OMAT and ODAT is show as Figures 7 and 8. When the scheduler works in the range of nominal separation of all medicines, the success rate it achieved is poor for n equal to 5 or larger. In contrast, the scheduler works better in the range of absolute separation of all medicines which is a wider range than nominal separation. We can also see from the figures that the advanced OMAT algorithm with MVF typically performs better than other algorithms. As a result, we will study it more depth in the future.

VII. Conclusions

In this paper, we presented the design ideas behind *Wedjat*, a medicine in-take reminder and monitor installed on a smart phone. This mobile computing application combines mobile phone based telemonitoring techniques with real-time scheduling algorithms to offer ubiquitous services to numerous out-patients. Notable accomplishments include the development of OMAT/ODAT medication scheduling algorithms and the implementation of an integrated mobile

VI. Related Work

As mentioned in the introduction, most attempts to reduce medicine administration errors have focused their efforts on developing "medicine dispensers". Most commercial products [16,17,18,19] are low cost, manual operating devices. A weakness shared by these devices is that their users must load medicine doses into these devices and then program their operation. Naturally, such a cumbersome operation is susceptible to human errors. Automated medicine dispensers [7,8,9,10] are definitely a step taken in the right direction. Some of these devices will be installed in the homes for the elderly or chronically ill. *Wedjat* represents a novel attempt to integrate healthcare support with mobile computing. As next generation smart phones such as iPhone™ and Google Phones™ become popular, such an application has the potential to reap a huge market share.

computing application. Plenty of work remains to be done after this initial effort: first and foremost, a thorough integration of *Wedjat* with electronic medical records (eMAR) and electronic personal health records (ePHR). Advance scheduling algorithms including those that can produce incremental changes to existing schedules should be investigated. The possibility of exercising on-the-fly changes of prescriptions in response to patient's conditions may also be explored.

REFERENCES

[1] Kohn LT, Corrigan JM, Donaldson MS (Ed.), *To Err Is Human: Building a Safer Health System*. Compiled by the Committee on Quality of Health Care in America, the Institute of Medicine, US National Academies; published by National Academy Press, 1999.

[2] *Preventing Medication Errors*. Quality Chasm Series compiled by the Institute of Medicine, US National Academies; published by National Academy Press, July 2006.

[3] Koppel B, et al., "Role of Computerized Physician Order Entry Systems in Facilitating Medication Errors," *Journal of American Medical Association*, Vol. 293, No. 10, 2005.

[4] Kuperman GJ, Bobb A, Payne TH, et al. "Medication Related Clinical Decision Support in Computerized Provider Order Entry Systems: A Review" Journal of American Medical Informatics Association, 2007.

[5] Baron RJ, Fabens EL, Schiffman M, Wolf E, "Electronic Health Records: Just around the corner? Or over the cliff?" *Annuals of Internal Medicine*, 2005.

[6] Wertheimer AI and Santella TM, "Medication Compliance Research," *Journal of Applied Research in Clinical and Experimental Therapeutics*, 2003.

[7] Wan D, "Magic Medicine Cabinet: A Situated Portal for Consumer Healthcare," *Proceedings of 1st International Symposium on Handheld and Ubiquitous Computing (HUC '99)*, September 1999.

[8] Murray MD, "Automated medication dispensing devices," Chapter 11 in *Making Healthcare Safer: a Critical Analysis of Patient Safety*, 01-E58, Agent for Healthcare Research and Quality, 2001.

[9] Governo M, Riva V, Fiorini P, Nugent C, "MEDICATE Tele-assistance System", *11th International Conference on Advance Robotics*. June 2003.

[10] Liu JWS, Shih CS, Tsai PH, Yeh HC, Hsiu PC, Yu CY, Chang WH, "End-User Support for Error Free Medication Process," Proceedings of High-Confidence Medication Device Software and Systems, pp. 34 – 45, June 2007.

[11] *Indivo*, the Personally Controlled Health Record system. http://indivohealth.org/.

[12] RSS: http://en.wikipedia.org/wiki/RSS_(file_format).

[13] Tsai PH, Yeh HC, Yu CY, Hsiu PC, Shih CSand Liu JWS, "Compliance Enforcement of Temporal and Dosage Constraints", *Proceedings of IEEE Real-Time Systems Symposium*, December 2006.

[14] Tsai PH, Shih CS, and Liu JWS, "Algorithms for scheduling multiple interacting medications," *Technical Report TR-IIS-08-001*, Institute of Information Science Academia Sinica, Taiwan, April 2008.

[15] Tsai, P. H., C. S. Shih, and J. W. S. Liu, "Algorithms for scheduling multiple interacting medications," Institute of Information Science, Academia Sinica, Taiwan, Technical Report TR-IIS-08-001, April 2008.

[16] e-Pill: http://www.epill.com/dispenser.html.

[17] MD2: http://www.epill.com/md2.html.

[18] Pill boxes: http://www.dynamicliving.com/automated_medication_dispenser.htm.

[19] Rx Showcase, http://www.rxinsider.com/prescription_dispensing_automation.htm.