DEPLOYMENT AND EVALUATION OF SELFPLANNER, AN AUTOMATED INDIVIDUAL TASK MANAGEMENT SYSTEM

IOANNIS REFANIDIS AND ANASTASIOS ALEXIADIS
Department of Applied Informatics
University of Macedonia, Thessaloniki, Greece

This paper presents SELFPLANNER, a deployed web-based intelligent calendar application that helps a user schedule in time and space her individual tasks. Contrary to other intelligent calendar assistants that concentrate on automating meeting scheduling, SELFPLANNER emphasizes on scheduling individual tasks and events, leaving meeting arrangement for external handling. The two key features of SELFPLANNER, also critical factors for its potential broader adoption, are problem modeling and user interface. SELFPLANNER supports simple, interruptible and flexible periodic tasks, arbitrary temporal domains, constraints over the parts of an interruptible task, binary constraints between tasks and preferences over their temporal domains, location references, classes of locations, time zones, etc. As for user interface, SELFPLANNER integrates with Google Calendar and a Google Maps based application, whereas it introduces an innovative way to define temporal domains, based on a combination of user-defined reusable templates and manual editing. The core of the system is based on the Squeaky Wheel Optimization algorithm, with efficient domain dependent heuristics. The paper is accompanied with an extensive evaluation of the system, comprising an analytic and an exploratory part. Evaluation results are very promising and suggest that SELFPLANNER constitutes a step towards the next generation of intelligent calendar applications.

Key words: Calendar applications, individual task modeling, scheduling, intelligent personal assistants.

1. INTRODUCTION

Traditionally, events and tasks are considered conceptually distinct aspects of a user's commitments and are treated separately by electronic personal assistants. Events usually have specific time and location reference and involve, explicitly or implicitly, other participants (e.g. a meeting, a class, an appointment with the doctor etc). On the other hand, tasks are mostly individual commitments having potentially a deadline to be accomplished (e.g. writing a paper or doing week's shopping); however, they are not given in advance a specific time window for execution, whereas location information is usually missing. This distinction is apparent in applications like MS-Outlook[1], where events are put directly into the calendar, whereas tasks are kept in a separate task list. In this way, the user always has a clear view of what event is coming next or how overloaded her agenda is, but hasn't a clear view of what task should be performed next or how many tasks she has to accomplish within e.g. the next week. Although MS-Outlook supports drag-and-drop operations between the task list and the calendar, thus transforming a task to an event, people usually does not perform this type of manual scheduling but prefer to create short-term schedules on-the-fly, just by watching the task list or the reminders. In this way, they frequently break the deadlines of their tasks or do not accomplish them at all.

Other applications concentrate either on events or on tasks only. For example, Google Calendar[2] and Yahoo! Calendar[3] handle events, whereas Chandler[4] focuses on collaborative tasks and information flow between users. None of these applications provides automated scheduling capabilities. A similar situation exists in research initiatives for managing personal time during the last two decades. As far as calendar management is concerned, research has concentrated for years on automating meeting scheduling, a widely agreed time consuming process (Berry et al. 2006; Modi et al. 2004; Singh 2003). On the other hand, task management initiatives usually concentrate on bookkeeping the completion status of a task, exchanging messages between users that have been assigned the same task (Conley and Carpenter 2007), and on raising reminders at appropriate times.

Our vision on managing individual tasks, i.e. tasks where a single person is involved, considers them in a unified way with events, by assigning to them event attributes, such as temporal domain, duration, location etc. Furthermore, we opt for a scheduling algorithm to reserve the necessary time for each task,

[1] http://www.microsoft.com/outlook/
[2] http://www.google.com/calendar/
[3] http://calendar.yahoo.com/
[4] http://chandlerproject.org/

in order to ensure its accomplishment, taking into account the user's stated preferences. In this way, the user can have a common view of her commitments, as well as a nearly-optimized short- and near-term schedule. Our vision has initially been expressed at (Refanidis, McCluskey and Dimopoulos, 2004).

This paper presents SELFPLANNER, a first step towards this vision. SELFPLANNER is a web-based application built on top of a fast domain-specific scheduler, and integrates several web technologies and Google-based applications. SELFPLANNER treats tasks and events equivalently and constructs locally optimized schedules using a rich problem model. On the other hand, meetings involving other people can be arranged using manual procedures of Google Calendar (or third party applications).

Apart from its scheduling engine, the two key features of SELFPLANNER are the rich problem model and the innovative user interface. Both of them are critical in order for the system to be adopted by its potential target group, i.e. people with very tight schedules (managers, academics etc). Concerning problem modeling, each task is characterized by a set of attributes such as duration, temporal domain, potential locations, periodicity, interruptibility etc. In case of periodic and interruptible tasks, additional information can be specified, such as the size of the period or the min/max duration of each part of an interruptible task. Ordering constraints between tasks, as well as several types of preference functions over the temporal domain of each task are supported.

Concerning user interface, SELFPLANNER uses innovative techniques to facilitate data entry. Perhaps the most tedious part of a task's definition concerns its temporal domain. This domain may consist of several temporal intervals distributed among large periods, e.g. store working hours. The user specifies temporal domains through a combination of template applications and manual editing. Templates are patterns that can be applied to large time periods. The user can create and reuse templates, as well as save her operations in order to reapply them to other tasks. Other user interface innovations concern a Google Maps based application that gives the user the option to define locations and compute temporal distances between them; classes of locations that provide alternative places where the user must be in order to accomplish a task; periodicity, where the instances of a task in different periods are scheduled separately; suggestions to relax constraints in case of overconstrained problems etc.

The rest of the paper is structured as follows: Section 2 defines the problem and discusses algorithmic issues concerning the adopted solution. Section 3 presents the SELFPLANNER system in detail, giving also illustrative use cases. Section 4 presents an extensive evaluation of the system, comprising an analytic and an exploratory part. Section 5 presents related work and, finally, Section 6 concludes the paper and poses future directions.

2. THEORETICAL BACKGROUND

In this Section we provide a rigorous definition of the problem of scheduling individual tasks and events, and give the key features of the underlying scheduling engine employed by the SELFPLANNER system (Refanidis 2007).

2.1 Problem Formulation

Time is considered a non-negative integer, with zero denoting the current time. There is a set T of N tasks, $T=\{T_1, T_2, \ldots, T_N\}$. Each task $T_i \in T$ is characterized by its duration dur_i [5]. All tasks are considered interruptible, i.e. they can split into parts to be scheduled separately. The decision variable p_i denotes the number of parts in which the i-th task has been split, with $p_i \geq 1$. T_{ij} denotes the j-th part of the i-th task, $1 \leq j \leq p_i$. For each T_{ij}, the decision variables t_{ij} and dur_{ij} denote its start time and duration. The sum of the durations of all parts of a task must equal its total duration (C1).

For each task T_i, the maximum and minimum allowed duration for its parts, $smax_i$ and $smin_i$ (C2), as well as the minimum allowed temporal distance between every pair of its parts, $dmin_i$ (C3), are given.

[5] In the following we use the notation x_i to abbreviate $T_i.x$, where x is any attribute of the task structure. In case of multiple subscripts, e.g. x_{ij}, the first one, i.e. i, indicates the task.

Depending on the values of $smax_i$ and $smin_i$ and the overall duration of the task dur_i, implicit constraints are imposed on p_i. For example, if $smin_i > dur_i/2$, then $p_i=1$, so task T_i is non-interruptible.

Each task i has its given temporal domain D_i, consisting of a set of intervals within them all of its parts have to be scheduled (tasks with infinite horizon of execution are not considered): $D_i=[a_{i1},b_{i1}]\cup[a_{i2},b_{i2}]\cup\ldots\cup[a_{i,Fi},b_{i,Fi}]$, where F_i is the number of intervals of D_i (C4). Obviously, $a_{ij}+smin_i\leq b_{ij}$ as well as $b_{ij}<a_{i,j+1}$ must hold for each $1\leq i\leq N$, $1\leq j\leq F_i$.

A set of M locations, $Loc=\{L_1, L_2, \ldots, L_M\}$ and a two dimensional matrix $Dist$ (not necessarily symmetric) with their temporal distances (non-negative integers) are given. Each task T_i has its own set of locations $Loc_i\subseteq Loc$, denoting alternative places where the user should be in order to execute each part of the task (the user has not to execute all the parts of a task in the same location). The decision variable $l_{ij}\in Loc_i$ denotes the particular location where T_{ij} will be scheduled (C5). All pairs of scheduled parts of tasks must have a minimum temporal distance equal to the temporal distance of their locations (C6).

For any subset of tasks $S\subseteq T$, a constraint c over these tasks may be defined, thus determining the valid ways to schedule the tasks of the set. Constraints refer only to time, not to location references; however the role of the locations in deciding when to schedule a task is important, since the decision to schedule a task at a specific location may affect the temporal domains. Each constraint c is defined by a function $propagate_c(S)$, which, given a set of partially instantiated tasks S, propagates the constraint $c(S)$ over the temporal domains of these tasks and returns *false* if any domain remains empty, otherwise it returns *true* (C7). As a simple example consider the ordering constraint over non-periodic tasks, denoted with $before(T_i,T_j)$, meaning that no part of T_j can start its execution until all parts of the T_i have finished their execution. In this case, $propagate_{before}(T_i,T_j)$ applies bounds consistency to D_i and D_j (Van Hentenryck, Saraswat and Deville, 1998).

Finally, a set V of time preferences over sets of tasks may be defined. A preference $v\in V$ over a set of tasks S is defined as a function $v:\prod_{Ti\in S}D_i\rightarrow\Re$, i.e. function v maps each combination of the temporal domains of the tasks of S to a real number. Preference functions are usually *max*-type functions that greedily try to estimate a best-case scheduling scenario based on the current domains of the involved tasks. For example, a unary preference could return the utility of the best time-window when the task could be scheduled, whereas a binary preference of an *away* type could return the utility of the maximum possible temporal distance where the two involved tasks could be scheduled. However, other types of functions, e.g. variations of expected utility, can be adopted as well.

So, after these definitions, the problem of scheduling individual tasks can be formulated as follows:

Given a set of tasks T with their attributes, a set of constraints C and a set of preferences V, find appropriate values for the decision variables p_i, t_{ij}, dur_{ij}, l_{ij}, where $1\leq i\leq N$, $1\leq j\leq p_i$ such as to maximize the expression:

$$\sum_{v_k\in V} v_k(S_k) \tag{1}$$

subject to the following constraints:

C1: $\forall i,\ 1\leq i\leq N:\ \sum_{j=1}^{p_i} dur_{ij} = dur_i$

C2: $\forall i,j,\ 1\leq i\leq N,\ 1\leq j\leq p_i : smin_i\leq dur_{ij}\leq smax_i$

C3: $\forall i,j,k,\ 1\leq i\leq N,\ 1\leq j\leq p_i,\ 1\leq k\leq p_i : j\neq k \Rightarrow t_{ik}+dur_{ik}+dmin_i\leq t_{ij} \vee t_{ij}+dur_{ij}+dmin_i\leq t_{ik}$

C4: $\forall i,j,\ 1\leq i\leq N,\ 1\leq j\leq p_i\ \exists k,\ 1\leq k\leq F_i : a_{ik}\leq t_{ij}\leq b_{ik}-dur_{ij}$

C5: $\forall i,j,\ 1\leq i\leq N,\ 1\leq j\leq p_i : l_{ij}\in Loc_i$

C6: $\forall i,j,m,n,\ 1\leq i\leq N,\ 1\leq j\leq p_i,\ 1\leq m\leq N,\ 1\leq n\leq p_m : i\neq m \vee j\neq n \Rightarrow t_{ij}+dur_{ij}+Dist(l_{ij},l_{mn})\leq t_{mn} \vee t_{mn}+dur_{mn}+Dist(l_{mn},l_{ij})\leq t_{ij}$

C7: $\forall c(S)\in C,\ propagate_c(S) = true$

2.2 Scheduling Issues

To solve the problem described in Section 2.1, we adapted the Squeaky Wheel Optimization (SWO) framework (Joslin and Clements 1999). The core of SWO is a Construct/Analyze/Prioritize cycle, as shown in Figure 1(a). Constraint variables are placed in a priority queue in decreasing order of an initial estimate of the difficulty to assign a value to each one of them. A solution is constructed by a greedy algorithm, taking decisions in the order determined by the priority queue. The solution is then analyzed to find those variables that were the "trouble makers". The priorities of the "trouble makers" are increased, causing the greedy constructor to deal with them sooner in the next iteration. This cycle repeats until a termination condition occurs.

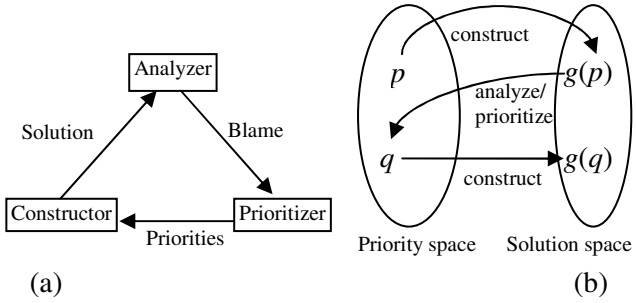

FIGURE 1. (a) The SWO cycle. (b) Coupled search spaces.

SWO is a fast but incomplete search procedure. As shown in Figure 1(b), SWO searches in two coupled spaces: The priority space and the solution space. The greedy construction algorithm defines a function g from the priority queues to the solutions, i.e. for each ordering p of the tasks, a schedule $g(p)$ is defined. However, function g may be neither surjective nor injective; so, many feasible solutions may not correspond to any ordering of the tasks in the queue.

We adapted SWO to SELFPLANNER using several domain dependent heuristics that measure the impact of the various ways of scheduling a specific task (including both time and location) to the remaining ones. In particular, the difficulty $diff(T_i)$ to schedule a task T_i is defined as the maximum between two metrics, m_1 and m_2, which in turn are defined as follows:

Metric m_1 of a task T_i is defined as the ratio between the total duration of the task and the net size of its temporal domain, i.e.:

$$m_1(T_i)=dur_i/net(D_i)$$

where the net size $net(D)$ of a temporal domain D consisting of a set of intervals is defined as the sum of the widths of these intervals. Metric m_2 of a task T_i is defined as the ratio between the minimum possible makespan of the task and the width of its domain, i.e.:

$$m_2(T_i)=min(makespan(T_i))/width(D_i)$$

The *makespan* of a task is defined as the distance between the start time of its earliest scheduled part and the end time of its latest scheduled part. Similarly, the *width* of a temporal domain is defined as the distance between the left end of its leftmost interval and the right end of its rightmost interval. Finally, the overall difficulty to schedule a set of tasks S is defined as the product of their individual difficulties:

$$overall(S)=\prod_{Ti\in S} diff(T_i)$$

So, tasks are initially placed in the queue in decreasing order of their individual difficulties, whereas each task is scheduled at the time slot where the overall difficulty to schedule the remaining tasks is minimized. For each possible time window to schedule the current task, constraint propagation is employed to update the temporal domains of the remaining tasks before computing their difficulties.

In case of preferences, a ratio between overall difficulty and approximated overall utility is minimized:

$$\frac{(overall(S))^a}{\left(\sum_{v_k \in V} v_k\right)^b} \quad (2)$$

Two key features of our implementation of SWO are worth mentioning. First, a task is considered trouble maker if there is no way to schedule it, such that after constraint propagation, all of the remaining tasks have non-empty domains. In other words, constraint propagation is performed before deciding when to schedule a task. In this way, infeasible scheduling options and failures are detected earlier. Second, trouble makers are promoted aggressively at the top of the priority queue. It has been found empirically that both constraint propagation and promotion to the head of the priority queue result in significant reduction in the number of iterations and the time needed to solve a problem. Finally, it has been shown that for this problem SWO with the difficulty heuristics is usually more efficient and effective (under time limit) than a complete constraint propagation search algorithm with the usual domain independent heuristics (Refanidis 2007).

3. THE SYSTEM

This Section gives a detailed presentation of the implemented system, i.e. its architecture, the key functionality points and some typical use cases.

3.1 System Architecture

SELFPLANNER is a web based application running over a planning server, which implements the SWO algorithm (Figure 2). All data are stored centrally, so the user can access the application from any networked computer. The user edits task data using user-friendly dialog boxes. The planning server solves the scheduling problem and inserts suitable entries in the user's Google Calendar account. Finally, the user watches her calendar directly into Google Calendar. The user can also add events directly into her Google Calendar account; during scheduling, these events are considered as busy time by the system. SELFPLANNER also integrates a Google Maps based application, thus giving the user the possibility to define a list of locations of interest, as well as to compute temporal distances between them. The Google Maps application "communicates" with the core SELFPLANNER system through a shared database.

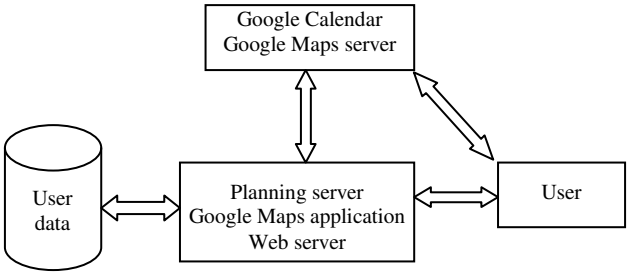

FIGURE 2. SELFPLANNER overall architecture.

SELFPLANNER encompasses several technologies and platforms. The main system consists of a Java application, with the user interface being a Java applet. All connections between the user and the system are secure. The Google Maps application uses PHP and Javascript. The planning engine that implements the SWO algorithm has been implemented in C++. Finally, user data such as task details and the current plan are retained as serializable Java objects (binary files).

3.2 Main Features

Figure 3 gives an overall view of the system, with the main windows open. The main entity of the system is the "task". Using suitable user interface modules, the user can define all the parameters of a task, as they have been described in Section 2.1.

Perhaps the most tedious job when defining a task is the definition of its temporal domain. SELFPLANNER adopts an innovative way for defining domains, based on combining manual editing and template applications. Manual editing allows the user to include or exclude (by clicking or dragging) specific time slots from a temporal domain (a minimum time slot of 30 minutes is assumed). On the other hand, templates allow the user to apply the same pattern of inclusions/exclusions over long periods. A template comprises a set of *green* and *red* values characterizing time slots over a relative interval. Three types of templates are supported: daily, weekly and monthly. The user can create and store her own templates. A template can be applied over the whole domain or over part of it, in four different ways: adding/removing the *green* slots, and adding/removing the *red* slots. The user can apply several templates, whereas the order of application matters.

Another innovation concerns the way temporal domains are retained in memory: they are not retained as lists of intervals but as lists of user actions. A user action can be either a manual addition/removal of a time slot in/from the domain, or the application of a template. The list of user actions is accessible to the user, who can modify it by changing the order of the actions or delete some of them. Efficient algorithms have been developed to answer questions such as whether a particular time slot is included or not in the domain.

SELFPLANNER treats all tasks as interruptible, with non-interruptible tasks being characterized by *smin=smax=dur*. The decision in how many parts to split an interruptible task is taken by the greedy scheduling algorithm. In addition, a task may be periodic. Periodic tasks are considered as collections of simple tasks. Each periodic task has a predetermined finite number of periods. The period may be either a day, or a week, or finally a month. The various instances of a periodic task are scheduled separately, so, depending on the temporal domain of the task, they may be scheduled in different ways. For example, the first iteration of a weekly periodic task might be scheduled on Monday, whereas the second iteration might be scheduled on Thursday. In addition, the user may ask not to schedule an instance of the task for specific periods (e.g. holiday breaks). Periodic task may be interruptible as well.

A task is characterized by a set of possible locations, i.e. in order to execute the task (or a part of it), the user must be in one of these locations. In case of interruptible and periodic tasks, different parts of the same task may be scheduled in different locations. In order to facilitate location entry, SELFPLANNER organizes locations into classes: A class is a set of distinct locations and it can be assigned to a task instead of a simple location. A location may be member of several classes. Travelling times are taken into account when scheduling tasks in different locations. Note that a special location, called ANYWHERE, may also be assigned to a task. This location has zero distance from any other real location, meaning that the user can be anywhere in order to execute the task (e.g. making a phone call).

Ordering constraints are also supported by the system, with *before*(*A*, *B*) meaning that no part of task *B* can start its execution before all parts of task *A* have finished their execution. Ordering constraint are also supported for periodic tasks, provided that both tasks are periodic and with the same type of period. In that case, the ordering constraint applies to all pairs of instances of the periodic tasks that share the same period. For example, if *A* and *B* are weekly periodic tasks, with *A* having iterations over weeks 1, 2 and 3 and *B* having iterations over weeks 2, 3, 4 and 5, then the ordering constraint will apply to the pairs of instances that will be scheduled in weeks 2 and 3. In case of an ordering constraint between a non-periodic and a periodic task, the constraint applies between the non-periodic task and the first or the last iteration of the periodic one.

Concerning preferences over alternative schedules, the system supports unary preferences over the temporal domains of the tasks. In particular, the user can specify whether she prefers the task to be scheduled as early or as late as possible (linear descending and linear descending preference functions),

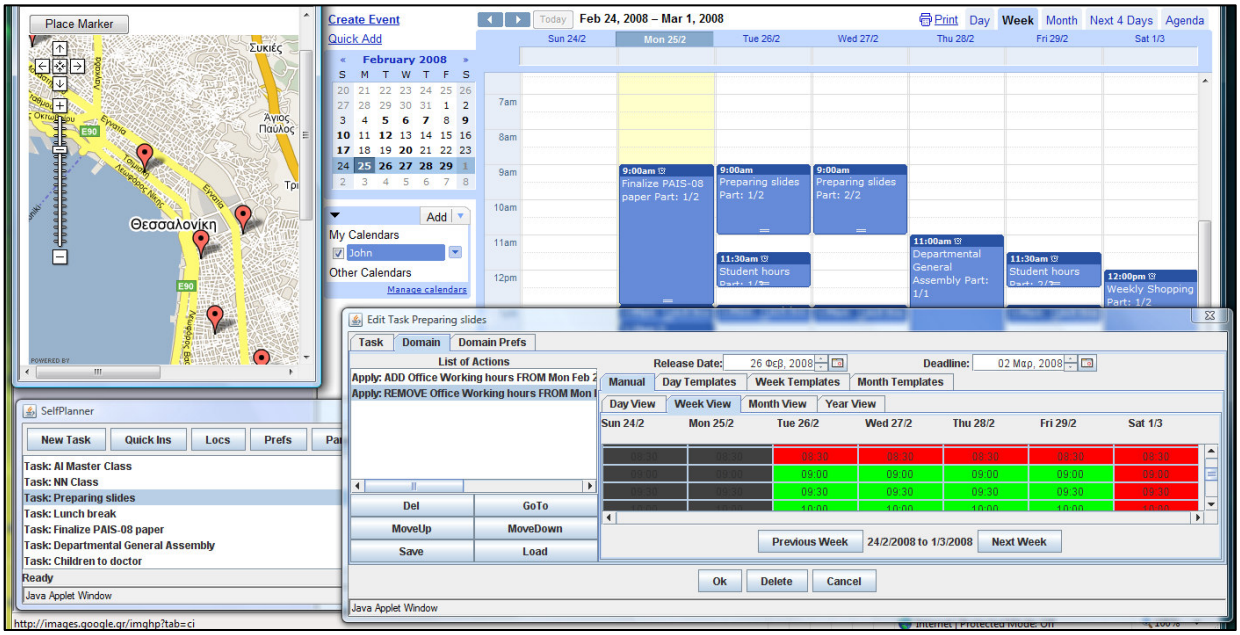

FIGURE 3. An overall view of SELFPLANNER system. The main application window is shown at the lower left corner, now listing the active tasks. The dialog box in the bottom right is the "Edit task" window, currently shown part of the domain of the task "Prepare Slides". In the upper left corner is shown the Google Maps based application for obtaining locations. Finally, in the upper right area, Google Calendar displays the user's current plan.

before or after a specific time point (step preference functions) or whether she is indifferent on how a task will be scheduled within its temporal domain.

SELFPLANNER also keeps track of the progress to complete each task. Unfortunately, currently this is not performed automatically, using some form of monitoring of the user's activities. So, the system asks the user at the login time as well as at the scheduling time, whether already scheduled tasks or parts of them, whose scheduling time has passed, have been accomplished or not. Accomplished tasks are removed from the task list whereas the durations of accomplished parts of tasks are subtracted from the remaining duration of their tasks. Tasks identified as non-accomplished are rescheduled.

In case of overconstrained problems, the system determines a maximizing subset of tasks that could be scheduled and updates the user's calendar with it. Maximization refers to the optimization quantity (1) (Section 2.1). The identification of this set is based on the best incomplete schedule that was encountered while trying to solve the scheduling problem using the SWO algorithm. Note that in order to achieve this functionality we changed the basic SWO cycle. In particular, each time a task or a part of task cannot be scheduled, the system attempts to schedule the remaining tasks using the same greedy construction algorithm. Then, the incomplete schedule is scored using formula (1) only for the scheduled tasks or parts of them. If eventually no complete schedule is found, the best incomplete schedule is retrieved and presented to the user. Furthermore, the user is notified about those tasks that have not been scheduled entirely, being asked to relax the domains and/or the constraints involving these tasks.

Finally, it is worth to note that SELFPLANNER supports time zones. Whenever the user edits a task, the temporal details of the task (release date, deadline, template applications, alternative views of the domain etc) are related to the time zone of the user's computer, thus rendering the system internationally operable. This feature was critical in order to open the system for public use (Section 4.2).

3.3 Use Cases

The following are some illustrative examples of tasks a user might want to insert in her calendar:

Case 1: Attend a performance in the theater. The performance will take place at the local theater, on Saturday evening, 21:00-23:00. This is the simplest type of task, since it is non-periodic, non-interruptible and is has a very precise time schedule and location reference. For such simple tasks, SELFPLANNER offers a *Quick Insert* functionality, in order to facilitate data entry.

One might suggest that such simple tasks could be added directly into Google Calendar. This option is supported by SELFPLANNER, however it is not recommended since it does not allow the user to specify precisely the location reference of the task, which is important in order to compute travelling times. Furthermore, tasks directly added in Google Calendar reduce the flexibility in case of overconstrained problems.

Case 2: Buying gifts for the children. This task employs two additional features of the system: The task has neither a specific time window to schedule, nor a specific location reference. Concerning time, the user should define the duration of the task, e.g. 3 hours, a deadline, e.g. before holidays, and the possible time slots when the task can be scheduled. The time slots might be when the shops are open, which can be selected by applying a weekly template defining the shops working hours.

Concerning the location reference, the user might be indifferent on where to buy gifts from. That is, there might be several malls in the city and the user might want to go to the nearest one wrt to other adjacently scheduled tasks. This is achieved by defining a class of locations, called e.g. Malls, that comprises all city's malls, and attach this class to the new task. While scheduling, SELFPLANNER will select a specific location from the class to schedule the task, having as criterion to minimize travelling time wrt adjacently scheduled tasks.

Case 3: Writing a paper for a conference. This is a typical example of a task with specific deadline. The novelty of this task is that it is interruptible. Apart from the deadline, the user estimates that she needs 30 hours in total to write the paper. She does not want to work for less than 2 hours and for more than 4 hours continuously, whereas her breaks should be at least 2 hours each.

Again a class of locations could be used to define plausible locations to write the paper, comprising e.g. home and office. The temporal domain of the task might be anytime excluding sleeping time, which can be defined using a suitable template. Finally, the user wants to write the paper as early as possible, so she attaches a linear descending preference model to this task.

Case 4: Teaching a class. This is a simple task like in Case 1, with the only difference that the task is weekly periodic. However, everything is well specified, i.e. the time-window is very precise within each week (e.g. every Thursday, 14:00-17:00), as well as the location reference (e.g. some University hall).

The procedure is similar to that described in Case 1, except that the user has to define the type of the period (weekly) and the number of repetitions (e.g. 13 weeks or provide a deadline). If the class will not take place at specific weeks, the user can easily remove these periods.

Case 5: Weekly shopping. This is a combination of Case 2 and Case 4. The task is non-interruptible, admits of scheduling, and is weekly periodic. To define its domain the user has to employ a weekly template with the shopping hours. Furthermore, the user might prefer to perform weekly shopping as close to weekend as possible, so she can assign a linear ascending preference model to the task. Note that preference models for periodic tasks are applied separately to each instance of the task.

Case 6: Preparing for the class. Before each class (Case 4), the instructor wants to devote 4 hours to recap the material and possibly revise it. This task can be performed in parts, with minimum duration for each part equal to 1 hour. Furthermore, revision should be completed before the class. To capture these interrelations, the user has to define this task as a weekly periodic one, with an ordering constraint between this and the 'Teaching a class' one (Case 4). So, within each week, all parts of the 'preparing a class' task should be scheduled before the 'Teaching a class' task.

4. EVALUATION

This Section aims at evaluating various aspect of the SELFPLANNER application. The evaluation comprises two parts: analytic and exploratory evaluation, with the latter concerning not only the current state of the system but also the general underlying idea of intelligent personal assistants.

4.1 Analytic Evaluation

We compared SELFPLANNER to Google Calendar, a well established calendar application. We selected Google Calendar for comparison for two reasons: first, it is also employed by SELFPLANNER and, second, it is a typical representative of many commercial calendar applications (e.g. MS-Outlook or Yahoo Calendar).

For the comparison we adopted the NGOMSL variation (Kieras 1988) of the GOMS model (Stuart, Moran and Newell 1983). We based the comparison on four tasks that can be easily performed with and without SELFPLANNER. While we measured the effort needed by the user for each task, we were not interested in the actual time (based on counting the number of keystrokes or mouse clicks etc) but in the number of simple subtasks needed to achieve each high-level goal. As "simple subtasks" we considered user actions such as "select a date", "open a dialog box", "change the current view" or "apply a template". In this way we estimated the cognitive effort required by the user to perform each task, something that is more tightly related to each specific approach, and not the actual time needed to perform each task, that depends on each specific implementation. Of course, defining what stands for a "simple subtask" depends on the user's experience; however, taking into account the target group of these applications and that any regular user may become experienced with them after some use, we decided to focus on above average desktop and calendar application users.

While designing the tasks, we had in mind that SELFPLANNER differs in nature from traditional calendar applications in that it employs a scheduler. So, the tasks that we have designed require limited scheduling that, in case of Google Calendar, can be performed manually by the user (we did not take this cognitive effort into account); otherwise the evaluation wouldn't be fair. Especially for SELFPLANNER, for the tasks where templates were used, we provide two alternative measures: one where the template(s) existed and one where the template(s) should be constructed from scratch using the system's default templates. Finally, we ignored locations references, since other systems do not support reasoning over them. A brief description of the tasks that were used in the analytic evaluation follows:

- *Task 1*: At Wednesday, February 4th, 2009, 10 am to 1 pm, there is a Board meeting at your Department.
- *Task 2*: The spring semester of 2009 starts with the first week of February and ends with the last week of May. You teach an AI class each Tuesday, 2 pm to 5 pm, excluding holidays (e.g. Easter).
- *Task 3*: At your university you usually have a 30 mins lunch break each weekday. The restaurant is open between 12:30 pm and 2:30 pm and you prefer to have the lunch as late as possible. Schedule all your lunch breaks for the 2009 spring semester (excluding holidays), trying to avoid overlapping with other commitments (e.g. your AI classes).
- *Task 4*: You want to write a paper within the first two weeks of 2009 spring semester. You estimate that you need a total of 30 hours work for this task. You neither want to work less than 2 hours or more than 4 hours continuously; your breaks should be at least 2 hours each. Your work has to be accomplished during your office hours.

Table 1 gives an overview of the effort needed to perform these tasks, with and without the use of SELFPLANNER[6].

[6] A detailed description of the actual steps for each system can be found at http://selfplanner.uom.gr/FourSimpleTasks.html.

TABLE 1. Comparison between SELFPLANNER and Google Calendar using the NGOMSL model.

Task	SELFPLANNER steps (without/with new templates)	Google Calendar steps
1	6	4
2	11	12
3	15 / 17	43
4	15 / 17	18

The four tasks used in our evaluation are typical representatives of the situations that might arise in practice while using calendar applications. So, Task 1 represents the simplest and most usual case, where a very specific event is added into a user's calendar. In that case, SELFPLANNER is marginally less efficient than Google Calendar, something that was expected since Google Calendar is a highly optimized application in terms of user interface. Task 2 concerns the introduction of a constantly periodic task, a feature that is also supported by all calendar applications. Although again no scheduling is required, SELFPLANNER takes advantage of its feature to allow removing specific periods at the time of task definition. On the other hand, Google Calendar requires that, after the task's definition, the user will move to the specific periods in the calendar and delete each instance separately.

Tasks 3 and 4 take advantage of the scheduling capabilities of SELFPLANNER. So, Task 3 concerns a fluent periodic task, where all lunches are scheduled at 2 pm (as late as possible), except for Tuesdays where the lunches are scheduled automatically at 1:30 pm. On the other hand, without SELFPLANNER the user has to go through all weeks and move the Tuesday lunches half an hour earlier manually. Finally, Task 4 concerns scheduling a non-periodic but interruptible task. In this case, SELFPLANNER splits the task into parts and schedules it within a week, trying to avoid classes, meetings and lunches. Without SELFPLANNER, the user needs to split and schedule each part of the task manually. Figure 4 gives a snapshot of the user's calendar for the first week of the semester, after having accomplished all four tasks using SELFPLANNER.

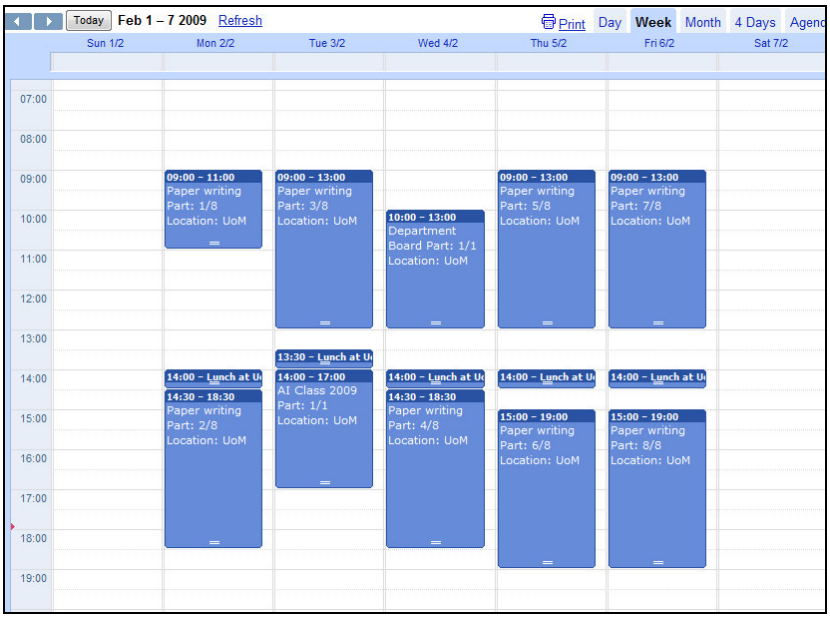

FIGURE 4. The user's plan for the first week of the 2009 spring semester, after accomplishing the four tasks of the analytic evaluation using the SELFPLANNER system. Note the start time of the "Lunch" event on Tuesday wrt the other days, as well as how "Paper writing" has been scheduled around the other events.

Note that the tasks used in the evaluation did not require rescheduling. However, if lunches were scheduled before classes (i.e. Task 3 were accomplished before Task 2), then rescheduling of the lunches would be required in order to schedule classes. In case of SELFPLANNER, this would be done automatically; otherwise manually.

The general result obtained from the analytic evaluation is that commercial calendar applications outperform SELFPLANNER in terms of user interface efficiency, and this is apparent when the introduction of a new event requires minimum scheduling. This was expected, since SELFPLANNER is still a research prototype, where attention is mostly paid on functionality than on user interface efficiency. However, even in this case, the predominance of commercial calendar applications is marginal. On the other hand, the more scheduling or rescheduling is required when inserting a new task, the more apparent the advantages of the scheduling engine of SELFPLANNER become.

4.2 Exploratory Evaluation

SELFPLANNER is an on-going effort that has started in 2007 and continues till now. In October 2008 the system reached a mature state that made it possible to open it for the general public. Furthermore, online documentation has been released. So, an open invitation to use the system has been sent to all major planning and scheduling mailing lists, resulting in more than 50 new registrations. In early December 2008 we asked all registered users to give us their feedback through a structured questionnaire[7]. We also gave them access to the four tasks used in the analytic evaluation; however, performing these tasks was not a requirement, since many users had used the system extensively, so they had enough experience to complete the questionnaire.

Using questionnaires is a common practice in software engineering to evaluate interactive systems, with several prototypes such as QUIS® (Chin, Diel and Norman 88) or SUMI® (Kirakowski and Corbett 93) being proposed over the years. Our questionnaire is closer to the QUIS® one, with the user being asked to express her feeling over several statements using an integer scale from 1 to 5, with 1 meaning "Strongly disagree", 3 meaning "Indifferent" and 5 meaning "Strongly agree". A "N/A" (no answer) option was also provided.

The questionnaire comprised 40 statements, which are grouped in three parts (in (Nielsen and Mack 1994) a number of questions between 20 and 40 is suggested for good questionnaires). The first part involved 25 statements concerning evaluating the system itself, such as "*Defining a new template for temporal domains was easy / intuitive*". This part gave us suggestions as to which parts of the system need further elaboration, in order for it to be more user friendly. The second part involved 10 statements concerning comparative evaluation between SELFPLANNER and any other calendar application the user might have experience of, e.g. "*SELFPLANNER gave me a better overall view of my workload*". The third part involved 5 statements concerning the user's overall attitude towards the underlying SELFPLANNER's idea, i.e. using a scheduler to arrange individual tasks, and the potential extensions of it. An example of these statements is "*Provided that scheduling technology is mature enough, I would accept an intelligent personal assistant that knows all my tasks/events and proactively (re-) organizes them*". Finally, there was a fourth part, where users were asked to suggest the three most important extensions to SELFPLANNER. They could select from a list of seven given potential extensions (e.g. "meeting scheduling" or "better scheduling engine") or write their own suggestions.

At the end of the evaluation period we had received n=17 questionnaires, however for specific statements the sample is smaller due to the "N/A" option. Due to the small number of questionnaires (n<30), we used the well known t-distribution and the independent one-sample t-test to verify the statistical significance of our hypotheses. Taking into account that the scale for the various statements is from 1 to 5, with m_0=3 corresponding to an "Indifferent" attitude, we used as null hypothesis the H_0: $m=m_0$, and we tested the hypotheses H_1: $m>m_0$ and H_2: $m<m_0$, where m is the mean value of the unknown

[7] Both the system and the questionnaire used for the exploratory evaluation can be found at http://selfplanner.uom.gr.

real distribution. Statistical significance coefficient was set to $a=10^{-3}$. The p-values for H_1 (or for H_2 when denoted) are given within parentheses.

All people were power users of desktop applications and above average programmers; however their prior expertise with calendar applications varied a lot. In the following paragraphs we summarize the results, trying to focus on the most interesting of them.

Concerning the first part of the questionnaire, people were satisfied by the Quick Insert functionality ($p<10^{-6}$) as well as by scheduling new tasks using existing templates ($p<10^{-4}$). However, they did not find it easy to define their own templates ($p\cong0,02$) or to modify the action list of domain definition ($p\cong0,06$). They also did not feel very confident with the whole temporal domain specification interface ($p\cong0,01$), although they significantly liked the whole idea of using templates to define temporal domains ($p<10^{-6}$)! These results can be interpreted as a suggestion for a more elaborated temporal domain definition interface (e.g. using wizards), however to some extent they can be ascribed to the limited time the users have worked with this innovative feature of SELFPLANNER.

Preferences over the temporal domains were well accepted ($p\cong10^{-5}$). People found intuitive the functionality for defining periodic tasks ($p\cong10^{-9}$) and for omitting specific periods ($p<10^{-5}$), with the latter being appreciated a lot ($p<10^{-8}$). People found the interface for defining interruptible tasks intuitive ($p<10^{-6}$), whereas they extremely liked the idea of supporting interruptible tasks ($p<10^{-13}$). They didn't find very intuitive the interface for defining ordering constraints ($p\cong0,005$).

Defining locations and attaching them to tasks was considered intuitive ($p<10^{-5}$ and $p<10^{-8}$ respectively), whereas they liked a lot the idea of reasoning over temporal distances between locations ($p<10^{-5}$). The interface for defining classes of locations was marginally considered intuitive ($p\cong10^{-3}$), however the general idea of organizing locations into classes was appreciated ($p\cong10^{-4}$). Online documentation was also considered satisfactory ($p<5\cdot10^{-4}$). Solving the scheduling problem and watching the resulting schedule on Google Calendar was considered easy ($p<10^{-5}$), whereas the resulting schedules were as expected ($p\cong5\cdot10^{-4}$). People liked a lot the interoperability with Google Calendar ($p<10^{-9}$). The system wasn't considered very stable ($p\cong0,007$) and the users wouldn't use it in its current form ($p\cong0,1$). However, the users liked the system very much and would use it in a more mature version ($p\cong3\cdot10^{-4}$).

Concerning the second part of the questionnaire, this has been completed by 12 people only. The users compared mainly to Google Calendar, however there were two references to MS-Outlook and one to Zimbra[8]. As expected, people hadn't considered SELFPLANNER competitive to existing calendar applications when entering simple tasks ($p\cong0,12$ for H_2) or simple periodic tasks ($p\cong0,05$). However, they preferred SELFPLANNER when omitting periods from periodic tasks ($p\cong6\cdot10^{-4}$), when scheduling fluent simple tasks ($p\cong10^{-7}$), when scheduling fluent periodic tasks ($p<10^{-5}$), when scheduling fluent interruptible tasks ($p<10^{-4}$) and when reasoning with locations ($p\cong10^{-7}$). It has not been shown that SELFPLANNER gave the users a better overall view of their workload ($p\cong0,002$), whereas the resulting plans weren't similar to what the user's would have created manually ($p\cong0,39$). However, the latter statement is a bit ambiguous, since not being similar does not mean being worse necessarily; perhaps this statement should be restated in a more precise way. Finally, and quite encouraging, the users claimed that they would consider to replace their existing calendar application with SELFPLANNER, or they would like to see features of SELFPLANNER being supported by their current calendar application ($p\cong0,0002$).

As for the third, and perhaps the most interesting part of the questionnaire, the users strongly liked the idea of embedding a scheduler into calendar applications ($p<10^{-10}$) and, provided that the scheduling technology is mature enough, they would accept an intelligent personal assistant to take control over their calendar and organize their tasks and events ($p\cong6\cdot10^{-4}$). They do not consider it a problem using a calendar application that runs over the internet, provided that the server is hosted by a trusted organization (or by the users themselves!) and the internet connections are secure ($p<10^{-4}$). Surprisingly enough, they are not very interested in using the system to automatically arrange meetings ($p\cong0,003$), perhaps because they want to keep control over their calendar. Finally, they were somehow reluctant ($p\cong0,11$) towards a

[8] http://www.zimbra.com/

system that knows their profile, collects information over the internet and inserts tasks and events into their calendar proactively (e.g. attending a concert) and accomplishes tasks automatically whenever possible (e.g. buying tickets).

Finally, the three most suggested features for future extensions to the system were improvements to the current interface and the underlying scheduling engine, and support for overlapping events.

The general conclusion we derive from this valuable evaluation procedure is that SELFPLANNER is a system with many innovative features that in many cases render it more efficient than existing calendar applications. On the other hand, it is also true that SELFPLANNER is a research prototype and cannot be compared with commercial calendar applications in terms of maturity or attractiveness. However, it is very encouraging that users are positive to adopt it for managing their individual tasks either in its current form or in a more mature version, and we are working continuously towards this direction.

5. RELATED WORK

SELFPLANNER is the first system that attempts to schedule the individual tasks of a user's task list into her calendar using constraint propagation and optimization algorithms. Furthermore, it introduces a new view of modeling this problem, including interruptible tasks, flexible periodic tasks, classes of locations, binary constraints and preferences; such constructs do not appear in traditional calendar applications. In this Section we concentrate on research prototypes that offer some form of automation for calendar applications.

There are plenty of systems developed over the last fifteen years that cope with the problem of automated meeting scheduling. Some of them concentrate on specific aspects of this problem (Garrido and Sycara 1995; Jennings and Jackson 1995; Sen and Durfee 1994; Sen and Durfee 1998). More recent efforts tend to incorporate learning components or to integrate with the Semantic Web. For example, RCal (Singh 2003) is an intelligent meeting scheduling agent that assists humans in office environments to arrange meetings. RCal agents negotiate with each other on behalf of their users and agree on a common meeting time that is acceptable to all users and abides by all the constraints set by all the attendees. RCal supports parsing and reasoning about semantically annotated schedules over the web, such as conference programs or recurring appointments (Payne, Singh and Sycara 2002).

CMRadar (Modi et al. 2004) is an end-to-end agent for automated calendar management that automates meeting scheduling by providing a spectrum of capabilities ranging from natural language processing of incoming scheduling-related e-mails, to negotiate with other users or making autonomous scheduling decisions.

PTIME (Berry et al. 2006) is an ongoing effort being developed under the CALO project (Myers 2006), that aims at facilitating meeting scheduling. The innovation of PTIME lies at its capability to learn the user's preferences thus adapting its future behavior, whereas it emphasizes in adopting natural language for interfacing with the user. Part of the work in the PTIME thrust is Emma, a personalized calendar management assistant designed to help its user handle email meeting requests, reserve venues and schedule events (Berry et al. 2008). Emma interfaces with commercial enterprise calendaring platforms and operates seamlessly with users who do not have Emma. It is designed to learn scheduling preferences, adapting to its user over time. The system is in initial deployment at SRI International.

Another effort within the CALO project concerns Towel (Conley and Carpenter 2007), an initial attempt towards an intelligent to-do list. Towel allows the user to organize to-dos (group, tag, check etc) as well as delegate them to other users or agents. Although to-dos can be seen as tasks, Towel emphasizes on to-dos manipulation rather than in solving the scheduling problem associated with actually performing these to-dos. Furthermore, it does not support all the advanced modeling features of SELFPLANNER.

6. CONCLUSIONS AND FUTURE WORK

SELFPLANNER is a deployed web based application which is constantly under development, with updated versions being uploaded several times a month. Its evolvement is driven both by user suggestions and by our overall vision.

As mentioned in the previous Section, most of the effort to add intelligence to calendar applications concentrated on automating meeting scheduling. We could imagine two explanations for this: First, people think that meeting scheduling is the most difficult and time consuming part of organizing a user's time. Although this might be true, meetings constitute a small part of our daily activities, whereas poor organization of the remaining tasks may result in significant waste of time and missed deadlines. Indeed, even scheduling together tasks with the same location reference, in order to avoid useless moves, would be of great value for many users.

On the other hand, we believe that the main reason why intelligent calendar systems do not focus yet on automated scheduling of individual tasks is due to a hidden consensus that it would be very difficult for a user to accept a machine-generated daily plan of activities (e.g. for psychological reasons among others). Our experience from using the system is that this is not absolutely true: Indeed, many of a user's tasks are very constrained, as for example giving a lecture or getting the children to school, so there is no need of scheduling for them. For the rest of the tasks, SELFPLANNER gives the user so many options to constrain the schedule and express her preferences, that it is very unlikely to get an unacceptable plan. In any case, our experience has shown that through the actual interaction with the system people learn personal policies of how to use it and get the most from it.

As for the psychological concerns, we can witness our evidence: SELFPLANNER has emerged as a way to fulfill personal organization needs. The initial motivation was to develop an intelligent calendar system able to schedule together tasks with the same location reference. However, through the actual use of the first prototype, several other needs came up, with many of them being already implemented and described in this paper. Interestingly enough, these claims are supported by our exploratory evaluation, where users said evidently that they like the idea of embedding a scheduler into calendar applications and, provided that the scheduling technology is mature enough, they would accept an intelligent personal assistant to take control over their calendar scheduler and organize their tasks and events! Furthermore, they do not have security warnings provided that the application is hosted by a trusted organization and secure connections are used.

As for the future, there are several directions in which SELFPLANNER could be extended. We are taking very seriously into account the results of the exploratory evaluation and work towards making the user interface more efficient and intuitive (especially the part that concerns the temporal domain definition). We also work on the underlying scheduling engine, in order to produce even better plans. Furthermore, our immediate future plans involve support for task priorities, overlapping events, multiple user calendars, editing of the resulting plan, as well as the development of a mobile interface for the client part of the system. Integration of meeting scheduling capabilities is also under consideration.

However, the most challenging extension concerns its transformation to a planning system. This next generation system will possess semantic knowledge about the user’s current state, a rich ontology with actions having preconditions and effects and a set of goals to be achieved. The system will help the user to insert tasks into her calendar in order to achieve goals or subgoals. For example, the user might set a goal for attending a conference, which could generate a sequence of actions being inserted into her calendar, including preparing the slides, bookings for the trip and travelling. Classical planning algorithms for causal reasoning can be used to solve the planning problem, probably in a mixed-initiative manner.

REFERENCES

BERRY, P., K. CONLEY, M. GERVASIO, B. PEINTNER, T. URIBE T. and N. YORKE-SMITH. 2006. Deploying a Personalized Time Management Agent. *In* 5th International Joint Conference on Autonomous Agents and Multi Agent Systems (AAMAS-06) Industrial Track, pp. 1564-1571.

BERRY, P., T. DONNEAU-GOLENCER, K. DUONG, M. GERVASIO, B. PEINTNER, B. and N. YORKE-SMITH. 2008. Emma: An Event Management Assistant. *In* ICAPS'08 System Demonstrations, Sydney.

CARD, S, T.P. MORAN and A. NEWELL. 1983. The Psychology of Human Computer Interaction. Lawrence Erlbaum Associates.

CHIN J.P., V.A. DIEHL and K.L. NORMAN. 1988. Development of an instrument measuring user satisfaction of the human–computer interface. *In* Proceedings of ACM/SIGCHI, pp. 213–218.

CONLEY, K., and J. CARPENTER. 2007. Towel: Towards an Intelligent To-Do List. *In* AAAI Spring Symposium on Interaction Challenges for Artificial Assistants, Stanford, CA.

GARRIDO, L. and K. SYCARA. 1995. Multi-agent meeting scheduling: Preliminary experimental results. *In* 1st International Conference on Multi-Agent Systems (ICMAS).

JENNINGS N.R. and A.J. JACKSON. 1995. Agent based meeting scheduling: A design and implementation, IEE Electronic Letters, **31**(5):350-352.

JOSLIN, D.E., and D.P. CLEMENTS. 1999. "Squeaky Wheel" Optimization. Journal of Artificial Intelligence Research, **10**: 375-397.

KIERAS, D. 1988. Towards a practical GOMS model methodology for user interface design. *In* Martin Helander: Handbook of Human-Computer Interaction. Amsterdam, The Netherlands: Elsevier Science Publishers, pp 135-157.

KIRAKOWSKI J. and M. CORBETT. 1993. SUMI: The Software Measurement Inventory. British Journal of Educational Technology, **24**(3):210–212.

MODI, P.J., M. VELOSO, S.F. SMITH, S.F. and J. OH. 2004. CMRadar: A Personal Assistant Agent for Calendar Management. *In* Workshop on Agent Oriented Information Systems (AOIS), New York.

MYERS, K. 2006. Building an Intelligent Personal Assistant. AAAI Invited Talk.

NIELSEN J. and R.L. MACK (ed). 1994. Usability Inspection Methods. Wiley.

PAYNE, T.R., R. SINGH, and K. SYCARA. 2002. Calendar Agents on the Semantic Web. IEEE Intelligent Systems, **17**(3):84-86.

REFANIDIS, I. 2007. Managing Personal Tasks with Time Constraints and Preferences. *In* 17th International Conference on Automated Planning and Scheduling Systems (ICAPS), Providence, RI.

REFANIDIS, I., T.L. MCCLUSKEY, and Y. DIMOPOULOS. 2004. Planning Services for Individuals: A New Challenge for the Planning Community. *In* Workshop on Connecting Planning Theory with Practice, Whistler, BC, Canada.

SEN, S. and E.H. DURFEE. 1998. A formal study of distributed meeting scheduling. Group Decision and Negotiation, **7**:265-289.

SEN, S. and E.H.DURFEE. 1994. On the design of an adaptive meeting scheduler. *In* 10th International Conference on Artificial Intelligence for Applications, pp. 40-46.

SINGH, R. 2003. RCal: An Autonomous Agent for Intelligent Distributed Meeting Scheduling. Technical Report CMU-RI-TR-03-46, Robotics Institute, Carnegie Mellon University.

VAN HENTENRYCK, P., V. SARASWAT, and Y. DEVILLE. 1998. Design, implementation and evaluation of the constraint language cc(fd). Journal of Logic Programming, **37**:139-164.