Kairoscope Coordinating Time Socially

ReeD Eric Martin

B.A. Human-Computer Interaction
Carleton College, 2003

Submitted to the
Program in Media Arts and Sciences,
School of Architecture and Planning,
in Partial Fulfillment of the Requirements for the Degree of
Master of Science at the Massachusetts Institute of Technology

September 2010

Author

ReeD Eric Martin
Program in Media Arts and Sciences
August 6 2010

Certified By

Henry Holtzman
Chief Knowledge Officer, Research Scientist
Thesis Supervisor
MIT Media Lab

Accepted By

Pattie Maes
Associate Academic Head
Program in Media Arts and Sciences

Kairoscope Coordinating Time Socially

by **ReeD Eric Martin**

Submitted to the
Program in Media Arts and Sciences,
School of Architecture and Planning,
in September 2010, in Partial Fulfillment
of the Requirements for the Degree of
Master of Science at the
Massachusetts Institute of Technology

Abstract

If everyone says time is relative, why is it still so rigidly defined? There have been many attempts to address the issue of coordinating schedules, but each of these attempts runs into an issue of rigidity: in order to negotiate an event, a specific time must be designated in advance. This model is inherently poor at accommodating life's unpredictability. Kairoscope looks at time from a human perspective, focusing on time as made up of a series of events, rather than simply a series of events in time. This removes our reliance on a fixed time system, thus allowing people to coordinate events socially and on the fly, without worrying about precision.

This thesis explores the creation of Kairoscope, rooted in ideas behind our perception of time, and created with the goals of reducing time-related stress, optimizing for use of time, and increasing social interaction. The proposal is a model of contextually-aware agents, constantly in communication with each other. The result is a socially-coordinated, constantly adapting, and highly malleable system to guide users through time and their schedules, without the heavy burden of precise planning. This thesis evaluates the potential implications of and the reactions to this model, as well as the design and interactions necessary to create such a system.

Thesis Supervisor: **Henry Holtzman**
Chief Knowledge Officer, Research Scientist
MIT Media Lab

Kairoscope Coordinating Time Socially

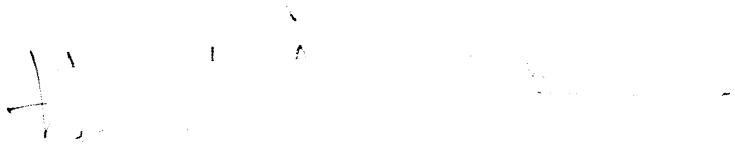
by **ReeD Eric Martin**

Thesis Reader

Pattie Maes
Associate Professor of Media Technology
Associate Academic Head
Program in Media Arts and Sciences, MIT

Kairoscope Coordinating Time Socially

by **ReeD Eric Martin**



Thesis Reader

Henry Lieberman
Research Scientist
MIT Media Lab

Acknowledgements

I am extremely grateful to all those who have helped make this work what it is today. The MIT Media Lab has served as a wonderful space for me to explore, research, and engage in conversations about how to shape the world in better ways.

In particular, I would also like to thank:

My advisor Henry Holtzman, who gave us the opportunity to shape and define the Information Ecology group, exploring different ways that information and our environments can permeate our lives, which ultimately led to this project.

All of the Information Ecology group, John, Matt, David, and Greg for being a great group of diverse folks who have always inspired and piqued my curiosities in new, sometimes bizarre, ways. Jeff, Agnes, Richard, and David Small in Design Ecology were always right there too for critique and inspiration. Our two groups served as a really great and unique combination of talents and personalities, which I'll always remember.

My thesis readers, Henry Lieberman and Pattie Maes, for providing simple and direct support through this process and helping to keep pointing me in the right directions. And Bill Mitchell, for always taking the time to support and inspire me.

My group-mates, Ned, Matthew, and Agata for working together on a project that ultimately served as great inspiration for this work.

Tara, who put up with my stress, lack of sleep, and constant overly-excited conversations about time and what it means. And Moxie, our wonderful cat who sat with me and kept me company when I stayed up late working on this project.

My family, who was always willing to discuss and critique my ideas and the bumps along the way.

I decided to drop everything and come to the Media Lab to push myself in new directions, be inspired by people from different backgrounds, push myself in ways that I wasn't comfortable with, and try to focus my energies on changing the world in a bigger way. I'm very impressed by this experience, research, and the people I've met along the way.

Table of Contents

Chapter 1

Introduction

"Clocks slay time... time is dead as long as it is being clicked off by little wheels; only when the clock stops does time come to life."

– William C. Faulkner,
The Sound and the Fury

We never seem to have enough time. There's not enough time to cook a real dinner, to grab lunch with that friend we've been meaning to for weeks, or to work on that project with the rapidly approaching deadline. Our lives are full of events: meetings, appointments, classes, and other

Figure 1. *Sculpture of the Greek God Chronos, the God of Time, by Hans Lott. Photo by*

Figure 2. *Bas-relief of the Greek God Kairos, the God of Opportunity, by Lysippos. Photo by*

scheduled and unscheduled activities. With the possible exception of corporate executives, there's almost always plenty of time in between, yet schedules don't match, timings vary, and events fail to materialize. We end up spending a large amount of time planning, scheduling, coordinating, or simply thinking about our upcoming activities instead of focusing on the ones we're engaged in presently.

The ancient Greeks had two words for time: Chronos (Kronos) and Kairos, represented by two gods of time. Chronos is, more or less, how we generally think of time today: a linear sequence (or chronology) of events. In this model, we become reliant on an endless linear flow of time. Kairos, although varying definitions and interpretations exist, generally refers to the "moment" or qualitative time: an idea that the only time that's important is *now* [1]. The concept of kairos was fundamental to the Sophists, where educated persons were described by Isocrates as "those who manage well the circumstances which they encounter day by day, and who possess a judgment which is accurate in meeting occasions as they arise, and rarely misses the expedient course of action." [2]

What if we spent less time caring about the specifics of how things will come to pass (scheduling and planning), and instead simply rest assured that they will indeed happen, freeing us to focus on the moment? This question forms the foundation of Kairoscope.

Of course, while it would be great to focus on the "now", it seems like we'd end up missing meetings, ignoring deadlines, and losing out on great opportunities. But what if some of these problems could be solved with technology, letting humans focus on the human aspect? Kairoscope looks at ways to remove the cognitive load of scheduling and managing time from the user, and instead relies on the combined scheduling power of participants and software. This allows a user to know that she is getting dinner with, for example, her friend John tonight at their favorite restaurant. When are they going? Why, at dinnertime, of course! (Or, more precisely, when they're both hungry and available.)

1.1 Background

In order to understand some of the goals and decisions behind Kairoscope, it's important to look at some of the sociological and psychological implications of how we manage and think about time today. Time perception itself can vary drastically from situation to situation: a boring class can seem to drag on forever, an exciting evening with old friends can make hours seem like minutes. But what senses are we using to perceive time? Is there a special sense of time, distinct from the other five senses, that allows us to perceive duration, or do we notice the passing of time through the perception of external stimuli and events?

Talking about the concept of time perception itself is odd, as we cannot precisely describe the passing of time without describing the events or changes in time. We experience each of these events as "present" when they occur, but the movement from experience to memory may be what allows us to construct an understanding of time having passed [3]. Ernst Pöppel describes 5 elementary time experiences: duration, non-simultaneity, order, past and present, and change or passage of time. While order is relevant to our understanding of time passing, multiple events that occur in close succession often result in incorrect memories of the order in which the events occurred. Yet an individual will often retain memories of "when" an event occurred: for example, in the morning, on a weekend, or in the summer [4].

In philosophy, there is a debate around the metaphysics of time elapsing, largely surrounding two aspects: presentism vs. eternalism, and objective time passing vs. temporal relations of precedence and simultaneity [5]. While there are differing viewpoints about how time itself passes, a human's perception of time can be considered to be grouped into 3 categories: past, present, and future [6]. This view, labeled "egocentric," is indeed indicative of how we organize events in our minds—relative to our present experience.

An individual's ability to accurately perceive the passing of time is variable, based largely on the experiences of events.

Normally, this variability would prevent us from being able to effectively manage large number of events in a day, particularly when the events require coordination with others. However, with the advent of coordinated time systems, we can constantly monitor and quantify the passing of time, based on a set of rules largely agreed upon around the world. Our time perception may be poor, but a linear sequence of numbers can help us properly quantify our understanding of time passing.

While this model is certainly effective at creating a baseline for time agreement, it's somewhat of a brute force solution. If our ability to perceive distance were similarly poor, a parallel solution would be to have a wristband giving our precise latitude and longitude at all times, allowing us to navigate physical space by these numbers. Certainly this idea seems absurd: navigating distance without factoring in relative distances and context would not provide a true understanding of a space, and could even lead you to walking right into a wall. A clock informs you that it's 9:23, without relating this to any temporal events in your day, even forcing you to calculate your own relative event spacing and durations ("How long do I have before that 11:15 meeting?") It is no wonder why we spend so much time "checking the time," to ensure we're not running through or missing our events.

Our time system, whether metaphysically accurate in its description of time or not, certainly is not how humans egocentrically perceive time. Just as we experience our physical world from an egocentric point of view ("That building is around 10 feet from me" rather than "That building is located at 48°51'22.55 N, 2°21'39.20 E, I am at 48°51'22.15 N, 2°21'39.15 E"), it is important to consider time as relative to our present.

1.2 Overview & Contribution

This thesis presents Kairoscope, a software system designed to help humans manage time and coordinate socially, without relying on numerical time systems. Kairoscope goes well beyond solving a scheduling problem, introducing an egocentric approach to navigating through time: guiding users through time from their own perspectives, focusing on the events that make up their lives. Through this novel approach to time perception and experience, a more clear sense of time emerges, opening up new possibilities for our abilities to coordinate, experience, and interact.

Kairoscope draws on human cognition to understand human time perception, psychology to look at human factors related to experiencing time, philosophy to explore concepts of time experience, social networking to place each user within their social context, artificial intelligence to understand user

habits and desires, and software agents to coordinate on behalf of users. By looking at the problem of time management and scheduling from a core human perspective, combining multiple diverse fields, rather than a simple technological problem solving angle, Kairoscope can take a radically different approach to managing events in time.

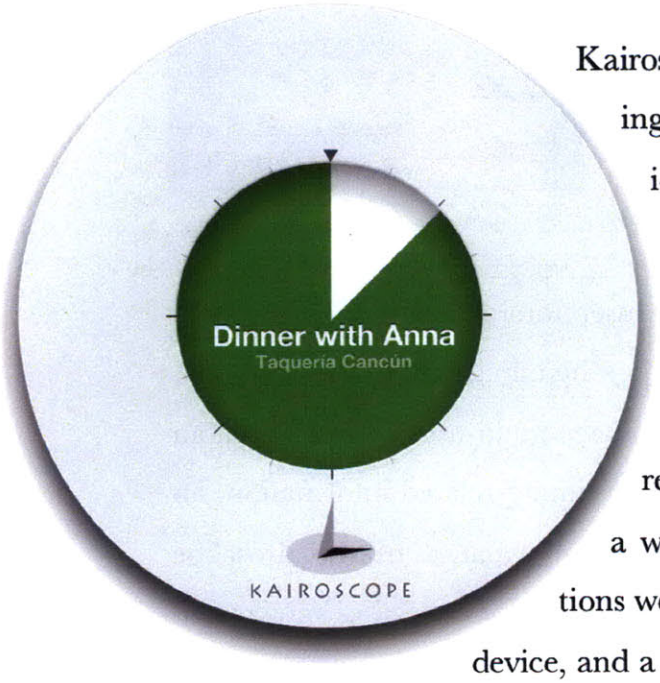


Figure 3. *Simple representation of a single upcoming event in Kairoscope, showing time remaining before an event, the type, location, and participants.*

Kairoscope acts as a guide through human time, acting on a user's behalf to represent time in a fashion that makes sense for each individual. This vision sees the ultimate replacement of clocks, representing non-contextual time information, with user-representative clocks *(fig. 3)*. There are a large number of potential devices and representations to be replaced, such as a watch or a wall clock. For this project, three implementations were created: a desktop clock, a clock on a mobile device, and a clock on a computer. Each of these clocks represents a user's personal time by showing the user his or her events. How much time is left before an event occurs, the amount of time an event occupies, the location of an event, and who is involved in an event are all represented, rather than any specific numerical times. Instead, the events are placed in context with each other and the user, showing relative distance from the present.

This project saw the creation of these three clock implementations and their user interface and design *(fig. 4)*.

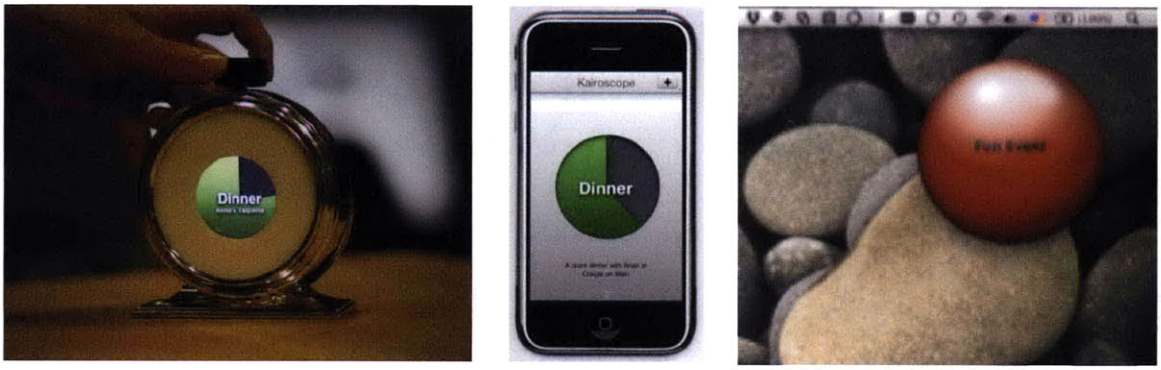


Figure 4. *From left to right, showing an upcoming event: a desktop clock, a mobile clock app, a computer clock.*

In addition to these direct user interaction implementation, the following were created: scheduling agent software that runs on mobile devices to manage multi-user coordination, an online database of aggregated timing-related information, an API to interact with this timing information from Kairoscope clients, an open online web presence for collecting known timing specifications like restaurant hours or event times *(fig. 5)*, and a micro-format for handling variable and egocentric timing information. In addition to the technical development, this thesis aims to describe and understand the psychology behind individual time perception and the factors that play a role in experience, time stress, and social interaction.

Figure 5. *Kairoscope Hours, online database for collecting known timing information such as a restaurant's hours, shown above.*

This project also oversaw the development of a basic user evaluation to gain initial feedback and understanding from individuals about Kairoscope. This was done in the form of

two sessions, one with particular tasks, the other designed for more general user feedback in the form of a survey. This evaluation helps to understand and shape future work and directions for Kairoscope.

1.3 **Scenario**

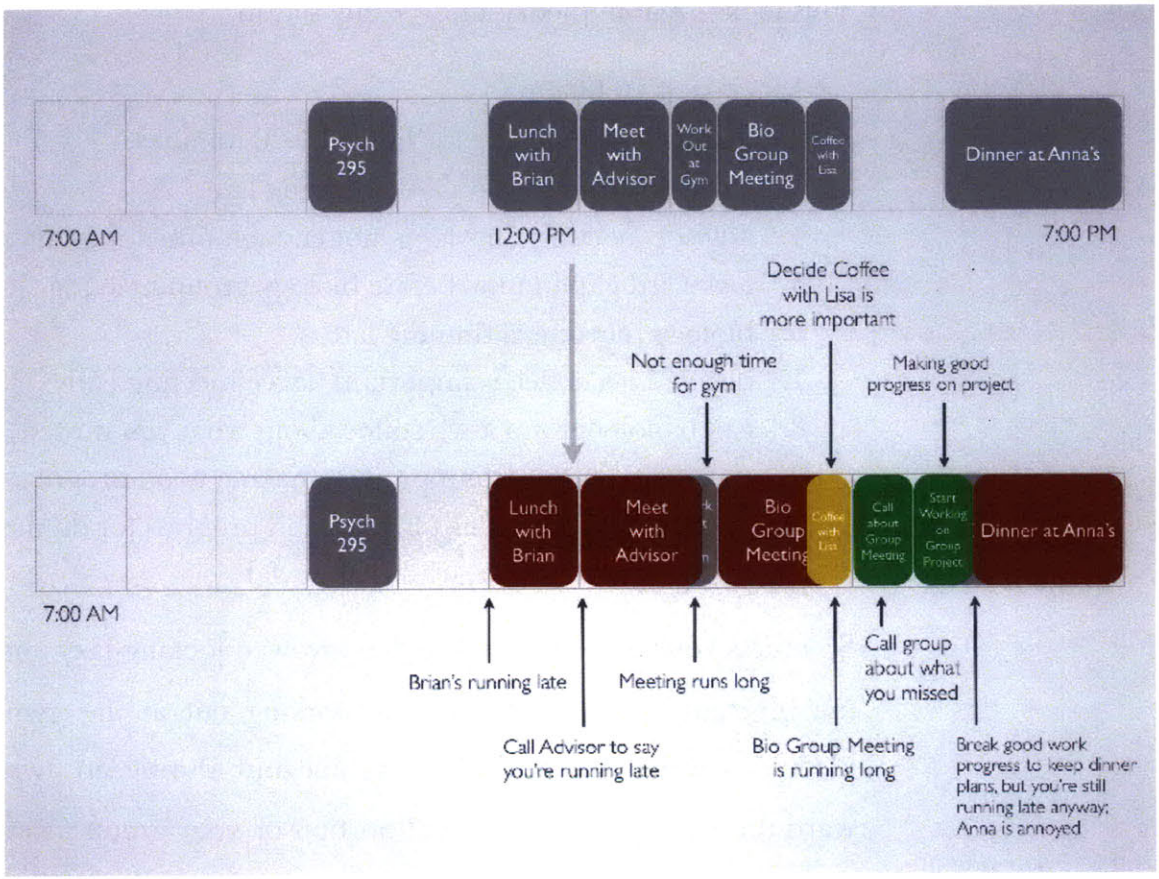


Figure 6. *Slide showing expected schedule (top) and actual schedule (bottom)*

Our schedules are constantly changing and our perception of time is highly variable, yet our schedules are built on a rigid time system. Imagine a day where you've tidily scheduled all of your events as follows *(fig. 6)*.

1. class at 10:00 am
2. lunch with Brian at 12:00 pm
3. advisor meeting at 1:00 pm
4. gym workout at 2:00 pm
5. biology group meeting at 2:30 pm
6. coffee with Lisa at 3:30 pm
7. dinner at Anna's at 5:00 pm

This looks great in theory, until reality sets in:

1. class at 10:00 am
2. Brian is running late for lunch by 20 minutes
3. call advisor to say you'll be running late
4. advisor meeting runs long, not enough time for gym
5. awkward gap in time before biology group meeting
6. biology meeting is running late
7. decide Lisa coffee is important, leave meeting early
8. call biology group after coffee about what you missed
9. start working on group project, making good progress
10. lose track of time, must break good progress for dinner
11. arrive late to Anna's, who is annoyed

So unless your to-do list for the day involved looking lazy and incompetent to your advisor, not working out at the gym, playing solitaire on your phone to kill time during an awkward timing gap, missing a vital portion of your group meeting, interrupting good progress on your project, and annoying Anna, it's unlikely the day turned out how you had hoped. In fact, handling all of these adjustments and changes may have added an additional layer of stress to your day, in addition to not accomplishing everything that was intended.

Kairoscope takes a fundamentally different approach to managing your day, redefining "time" based on your events, not the other way around. In this scenario, as soon as Kairoscope recognizes Brian is running late for lunch, a series of actions is performed adjusting the timing of virtually every other event in the schedule. For example, the advisor meeting may automatically be adjusted back slightly, working out at the gym could be bumped to after coffee, the group meeting could be moved up to coincide with the end of the advisor meeting, coffee with Lisa can be adjusted as you realize the group meeting is running a bit late, and when you begin making good progress on the project, dinner can be pushed back.

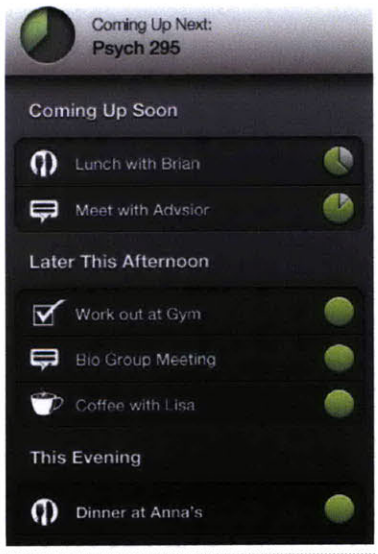


Figure 7. *Kairoscope Mobile App view of relative, non-time-precise upcoming events in a day*

While much of this can happen through some basic intelligence on the part of the scheduling agent, one of the fundamental elements that allows for greater flexibility is the idea of

removing time precision. Instead of knowing the precise times for when events will occur, your day's schedule could instead appear relative to your present, with a visualization showing relative amounts of time remaining, grouped by common mental time representations *(fig. 7)*.

One of the major advantages of such a model is in fact that, should times vary slightly, this information does not necessarily need to be exposed to the user until it will have an impact. For example, it may become clear at 11:00 AM that Brian will be running late to lunch. At this point, Kairoscope can begin to reschedule the advisor meeting to occur 15 minutes later. Assuming the event still fits within the advisor's schedule, since it is still occurring this afternoon, the advisor may not even be aware that the meeting was pushed back slightly—preventing the uncomfortable situation of having to run late to the appointment. If, however, there is a last-minute change, such as the biology meeting running late, a direct adjustment of the coffee appointment is relayed to Lisa in real-time, notifying her of the adjustment and offering the option to agree or reject the change.

1.4 Thesis Organization

After the introduction, this thesis is organized into six sections: Goals & Motivations, Related Work, Description, Design, Evaluation, and Conclusion.

Goals & Motivations lays out the reasoning behind Kairoscope, the primary areas that are affected, and the goals of the project.

Related Work describes current and past research and work that has been done in similar spaces. This section provides a basis for comparison between implementations and goals.

Design looks at the method with which Kairoscope is conceived and how the project's goals are approached.

Implementation showcases the design and implementation of the Kairoscope model, particularly describing the technical underpinnings and the user interaction models.

Evaluation discusses a user study and feedback performed to gain insight into how individuals might use, understand, and react to Kairoscope.

Conclusion lays out the future vision of Kairoscope as a model for time representation, providing a summary of what was designed and learned, describing current concerns and limitations, and looking at future directions.

Chapter 2

Goals & Motivations

"I am tired of the imposed rhythms of men,
Tethered time, restrained and trained
To a monotonous beat
Digital time blinking exactness
Unliving."

– Phillip Pulfrey
"Conjecture," Beyond Me

There are three primary goals of Kairoscope that are investigated: reducing stress caused by chronic time pressure, improving time management through optimizing for time efficiency, and promoting happiness by enabling opportunities for social interaction. The motivations for targeting

these goals are inspired by human time perception and time management: minimizing the time spent on activities that cause excessive difficulty or stress while maximizing for enjoyment of life and increased human interaction.

2.1 Chronic Time Pressure

Time pressure has become an increasing social problem affecting quality of life, stress, and overall health of individuals in modern societies, despite increases in quality of life in a variety of other aspects, including subjective well-being [7]. While there are a variety of possible explanations for this paradoxical relationship between increasing living standards and increasing time pressure, the fact remains: as a society, time increasingly feels like a stress that is weighing us down .

A few methods of addressing this negative trend of time stress have emerged, at least from a societal angle, ranging from imposing mandatory work day hour limits to simply managing one's tasks better. Countless articles have been written, support groups have been created, and seminars have been offered, suggesting new ways to improve "time management," often by attempting to take better advantage of time by decreasing cognitive load of remembering specific tasks and due dates (such as "Getting things done" or GTD). There may be certain successes with models such as this, yet they inherently seek to alleviate time stress through adding an

additional layer of complexity in managing one's stressors. At the same time, an untold number of newspapers and magazines publish articles encouraging people to "slow down" and enjoy the "little things" in life—how can we be expected to manage everything we want to do with everything we need to do, all while slowing down and enjoying the small details?

It has been suggested that one of the primary aspects of the perception of time shortage is the direct result of feeling rushed [8]. It therefore seems logical to look at scheduling and managing our time in a way that does not cause us to feel rushed. This, in turn, may reduce the amount of chronic time pressure we experience. Often it is the quantity of activities in a given day, rather than the number of hours spent in activities, that causes additional stress [9], as an individual is forced to manage and schedule each of these events. For example, two 3-hour meetings would generally cause less time-related scheduling stress than twelve 30-minute meetings, yet would occupy the same amount of actual time spent in meetings. If we can remove a certain level of burden on the scheduling process for users, while increasing flexibility, there may be a possibility for lowering some of the perceived rushing, thereby reducing chronic time pressure.

In addition, there have been a number of studies looking at our perception of time and what causes certain moments to seem as if they drag on forever, while others feel much shorter

than they actually are. Mihály Csikszentmihályi describes the idea of *flow* as, "the state in which people are so involved in an activity that nothing else seems to matter. [...] being completely involved in an activity for its own sake. Time flies." [10] It's apparent that the amount of time one spends in *flow* can be somewhat inversely related to the amount of time pressure one feels. While a reduction in time pressure does not have a direct causal relationship to the concept of *flow*, one of the characteristics of an individual experiencing flow is a lack of awareness of time in an absolute sense [11]. By creating a system that does not demand a continual awareness of time progression, opportunities for engaging in activities that reduce time pressure are increased.

2.2 Optimizing for Time Efficiency

Along with chronic time pressure, there is an inherent difficulty in crafting schedules that can optimize for time efficiency. While there are full-time jobs dedicated to the task of efficiently managing the schedule of another person, this is currently restricted to certain, generally work-related, positions: one could hardly imagine a world where 50% of the population is charged with efficiently managing the other half's schedules. (And, perhaps needless to say, who would manage the schedules of the former?)

Removing the burden of time precision from individuals has the benefit of also being able to more effectively optimize time through a layer of technology. This optimization can occur in a few fundamental ways: First, events can simply be grouped together by time, with the goal of minimizing awkward gaps. While the definition of awkward gaps may vary from person to person (and by location), the goal is to avoid situations where a gap between events is long enough to feel annoying but too short to engage in another activity in the interim.

Second, schedules become easily adaptable when one event changes or is cancelled, again limiting extra space that might occur as a result of changes. Traditionally, if an event is cancelled, one can often be left with an awkward moment in a schedule while waiting for the next event to occur. If the time precision is removed, there is more flexibility to adjust the subsequent event with potentially minimal (or no) consequences. This can potentially allow, for example, a 30-minute cancelled meeting to automatically bump up the following meeting if the impacts are minimal or beneficial.

Third, additional timing abilities are opened up when events aren't limited to standardized fixed durations, start, or end times: a 24-minute meeting can be accounted for just as well as a 30-minute one. One benefit of this is that it allows repetitive events to become optimized for true durations. The

meeting that always runs short can now more accurately allow for subsequent meetings to begin earlier, without regard for rounded timing.

Finally, while not explored directly in this thesis, optimizations can be performed to group events based on location, participants, or type. The abstraction of time precision allows for events to be grouped by concepts other than chronology.

While Kairoscope looks specifically at the optimizations performed based on event timing, this model opens up the possibility for additional increased knowledge and intelligence in managing a schedule to optimize more effectively. For example, it could be possible to tap into GPS data for an upcoming train's timing to facilitate ending a meeting with enough time to precisely catch the train to dinner, thereby minimizing time spent waiting for a train, and coordinating the dinner across multiple individuals to adjust its time to fit with the expected arrival time. Kairoscope is designed in such a fashion that additional points of information and relevance can be integrated to provide additional scheduling assistance, based on the requirements or desires of each participant.

2.3 Optimizing for Social Interaction

While optimizing for efficiency clearly can enable new opportunities, efficiency itself is not always an interesting metric, particularly when developing a system designed to affect hu-

mans. In fact, it can often be the contrary: efficiency optimizations have the potential to simply yield even more opportunities to pile onto one's schedule. It is therefore important to think through how we're interested in optimizing time usage, and for what purposes. In order to reduce chronic time pressure, it behooves Kairoscope to choose optimizations wisely. As such, while optimizing for time efficiency remains an integral component of how Kairoscope works, optimizing for social interaction provides an example of one of the benefits of such a system: optimizing for humans.

Causes & Problems

While technological advances in communication have enabled new methods of interacting, facilitated long-distance communication, and offered increased efficiency in managing large numbers of interactions, they have also shown to potentially be decreasing the number of face-to-face interactions *(fig. 8)* [12].

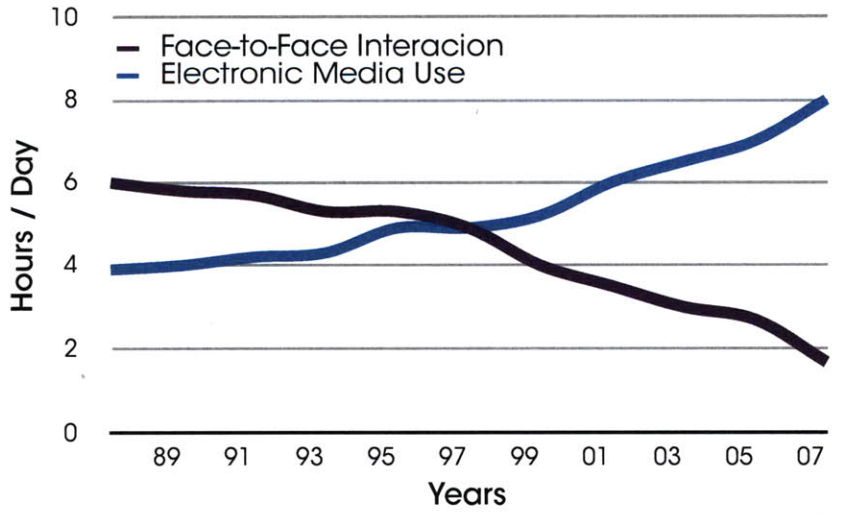


Figure 8. *Graph of Face-to-Face Social Interaction vs. Electronic Media Use showing decreasing face-to-face interactions. A. Sigman, 2009.*

A commonly cited complaint of mid-twenties through mid-fourties individuals in America has been a lack of an easy way to engage with their friends face-to-face.

Another contributing factor is the design of our cities and towns. Urban development, particularly since the inception of the automobile, has not necessarily designed with humans in mind. In fact, to the contrary, our cities have seen an increase in sprawl and a higher reliance on car-based cultures. This has literally increased the physical distance between people in a city, thereby reducing many opportunities for social interaction, particularly outside of organized activities, such as work or school [13].

There have been numerous studies looking at overall stress levels compared with levels of social interaction and social isolation. The seminal 1976 Sidney Cobb article, "Social support as a moderator of life stress," discusses the physiological and behavioral implications of social interaction, although framed within the context of social support [14]. While social interaction does not necessarily imply support, clearly increasing social integration opportunities can easily pave the way to stronger social bonds and support. In turn, increased social interaction, particularly face-to-face [15], does indeed lead to self-reported happiness levels increasing and stress levels decreasing, in addition to a variety of physiological improvements. In fact, it has been shown that an increase in positive

face-to-face social interactions can increase cortisol and oxytocin levels in humans [16, 17]. While Kairoscope certainly cannot create positive social interactions (this is a uniquely human ability), it can facilitate the frequency of occurrence of interactions.

Proposed Methods

There are therefore two main methods in which Kairoscope attempts to increase social interaction. First, the model of reducing scheduling difficulty stands to also facilitate the scheduling of social events in one's life, where the burden of planning is largely alleviated on the part of each individual. "We should hang out sometime" may not be restricted to a statement which is said, and meant, but often not realized due to the weight of scheduling the actual event. Kairoscope can follow through on these lightweight commitments proposed between individuals.

Second, Kairoscope has the ability to increase opportunities for unscheduled social interactions. As Kairoscope gains information about each individual user's schedule, the schedules can be compared within a social network to look for similar opportunities and potential overlaps. A simple example of this would be a casual coffee between friends: On Monday, Paul and Emilie decide to get coffee "later this week" at Café Flore. A group of three of their friends are already planning to get coffee on Thursday at the same café. Kairoscope can

add additional probability to this time on Thursday for Paul and Emilie's coffee such that, assuming there are no other conflicts, Kairoscope will be more likely to schedule their coffee to overlap with their friends.

This lightweight approach to unspecified social opportunities increases the likelihood an interaction may occur, without adding additional stress, planning, or even requiring any interaction. Should last minute changes affect Paul and Emilie's coffee, there is no negative impact—no direct plans were ever made. This creates a sense of social serendipity, increasing the frequency individuals might encounter friends, without requiring specific plans to be made.

Indeed, certain variables need to be taken into consideration to avoid also increasing unwanted social interactions such as, for example, running into your socially embarrassing friends during a business lunch meeting. Since these interactions can certainly already happen in one's daily activities, the current implementation does not concern itself too much with *avoiding* social interactions, although one could easily see a corollary to the goal of social serendipity: social avoidance.

Chapter 3

Related Work

"How long a minute is, depends on which side of the bathroom door you're on."

– Zall's Second Law

There is a wide body of related work in the areas of time management, schedule management, artificial intelligence planning, intelligent and cognitive load-reducing agents, and commonsense scheduling. Some particularly relevant work includes EventMinder [18] and CMRadar [19], which evaluate using agents to manage calendars and sched-

uling. EventMinder's goal-oriented approach seems particularly relevant in the context of Kairoscope. CMRadar is part of a larger project to help coordinate and manage users' lives, looking specifically at the negotiations and optimization models necessary to facilitate scheduling between users. P. Maes' defining work on software agents that represent and act on users' behalves, including "Agents that reduce work and information overload" [20] and "Learning interface agents" [21] are fundamental to the understanding of how agents can be treated as personal assistants and can learn from a user's habits and decisions over time. In addition, K. Sycara and D. Zeng lay out how more than one agents can interact together to negotiate and build consensus in "Coordination of multiple software agents" [22].

There has also been a reasonable amount of work done in calendar optimization agents, such as Jennings and Jackson's "Agent-based meeting scheduling: A design and implementation" [23] or Garrido and Sycara's "Multi-Agent meeting scheduling: A design and implementation" [24] which describe early agent-based scheduling agents. In addition, the more context-sensitive "Learning User's Scheduling Criteria in a Personal Calendar Agent" by Lin and Hsu show a model for learning a user's needs and desires over time, without needing to specifically ask [25]. There are also some novel approaches to schedule coordination, such as Kim *et al.*'s

"Compensatory negotiation for agent-based project schedule coordination" which looks at how agents can negotiate when each has distinct constraints and needs, weighing impacts through a model of compensation [26]. In this model, agents that are forced to compromise, accepting less-than-ideal situations, are given compensation by other agents involved. This of course forces an agent to consider the potential compensatory impact on demanding a situation which may be disadvantageous to another party.

While these models help to think about and set the stage for Kairoscope's implementation, there are a few primary aspects of Kairoscope that stand out from the approaches that exist:

1. Kairoscope hypothesizes that the specificity of time is not relevant to managing a series of events in a user's day, and that by removing the reliance on time precision, we can both more effectively schedule and continually adjust times for events in a day, allowing users to focus less on times and more on activities.

2. Kairoscope does not assume events are always fixed. Events may have varying levels of flexibility, and that flexibility should be exposed and used to easily modify on-the-fly, to best take advantage of moments and situations as they occur.

3. Kairoscope is designed to increase social interaction by coordinating events and locations of its users socially, crowdsourcing an individual's social calendar through a social network.

3.1 Schedule Management

There are a variety of well-known commercial and open source calendaring systems, including Microsoft Outlook [27], Google Calendar [28], MeetingMaker [29], Lotus [30], iCal [31], and CalDAV [32]. Each of these employs slightly distinct strategies and protocols, but each is ultimately built on the same fundamental idea of users managing sets of events in time and sending invitations to coordinate events with others. Some of this software is aware of each user's availability, and is able to point these out to someone who is attempting to schedule an event. A few offer the ability to ask for "next available meeting slot," overlaying the availability data for all invitees and looking for gaps in scheduled events *(fig. 9)*.

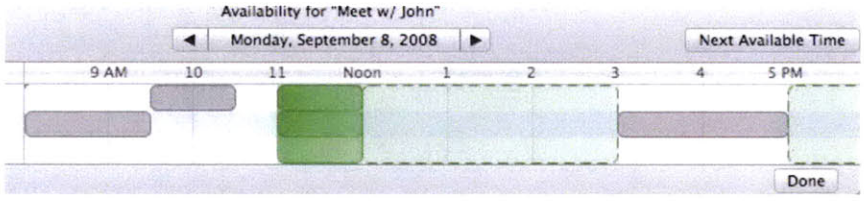


Figure 9. *Availability panel in iCal, showing time slots where availability exists for both parties*

The strategies employed by these calendaring systems build off of traditional calendars and datebooks, which were tradi-

Figure 10. *Classic datebook carried around, filled in by pen on paper. Photo by Jenni Ripley.*

tionally filled in by pen on paper *(fig. 10)*, carried around—particularly among those who had jobs with many meetings. The natural progression to moving from pen and paper to screens took place, with a few obvious benefits and changes over the years (networked invitations), but haven't fundamentally changed concept since the first calendaring systems appeared on computers in the late 1970s. As such, these calendaring systems suffer from the same problems Kairoscope is trying to avoid: they're stuck in a rigidly-structured time system, with no fundamental understanding of their users. In addition, the scheduling software is divorced from how a user interacts with time during the day, relying on the individual to remember what events occur at what times.

3.2 Time Management

Time management is an area that has largely focused on methods and strategies, with software designed to help users follow these methods. For example, "Getting Things Done" or GTD is not a scheduling system at all, but rather a method for organizing and managing tasks during one's day. While Kairoscope does not currently directly address accomplishing non-event tasks, GTD's concept is based on the principle of a "RAM dump," whereby a user writes down externally what they need to do, as a method of relieving their cognitive load [33, 34].

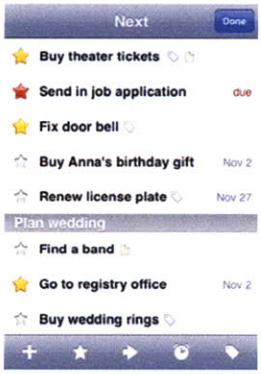


Figure 11. *Things : a GTD-based task manager for organizing and completing tasks.*

There have been a variety of software tools modeled after this concept of organizing task lists and methods for completing them, including iGTD [35], OmniFocus [36], ThinkingRock [37], and Things [38] *(fig. 11)*. While virtually all of these software tools are solely designed around organizing your tasks, it's interesting to think about this concept when applied directly to events and time. Writing down all of your engagements into a datebook can shift the burden of event management from an individual's memory to a piece of paper (or software). However, this memory dump is only as useful as the relevance of the information recorded. If event modifications or rescheduling are common, or if events frequently don't match up in real-world timing to that which was recorded, the organization becomes less relevant and can introduce additional levels of stress when one party is forced to wait for another.

3.3 Intelligent Planning & Scheduling

There are a few software projects that have been created with the goal of assisting a user to make plans more easily. The most closely related to Kairoscope are CMRadar, EventMinder, RhaiCAL [39], and PTIME [40]. Each project takes a different approach and presupposes that we work within the standard constraints of regular time system, but there are many interesting aspects worthy of considering in the context of Kairoscope.

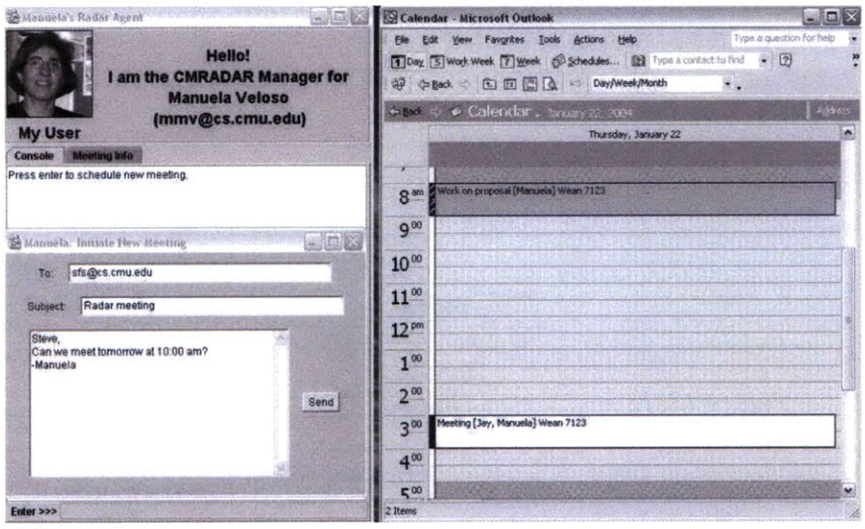


Figure 12. *CMRadar scheduling agent (left) interacting with a user's MS Outlook schedule (r)*

CMRadar is a project from Carnegie Mellon designed as a "personal assistant agent for calendar management." CMRadar is an end-to-end automation system for scheduling, including autonomously making scheduling decisions and negotiating with other users on your behalf. CMRadar is particularly concerned with the personal assistant aspect of scheduling, acting on the user's behalf for managing scheduling issues, while using standard scheduling software such as MS Outlook *(fig. 12)*. This approach was particularly interesting to see how the project handled system constraints and generated feasible meeting times. Some of the same ideas were employed in the design decisions for Kairoscope. A unique feature to CM Radar seems to be the use of a "predicted cost of negotiation" for having to work with another user/agent to make scheduling decisions. Since Kairoscope assumes shared goal agents, this was largely replaced by con-

fidence predictions, although there could be room in the future to evaluate negotiation costs as well.

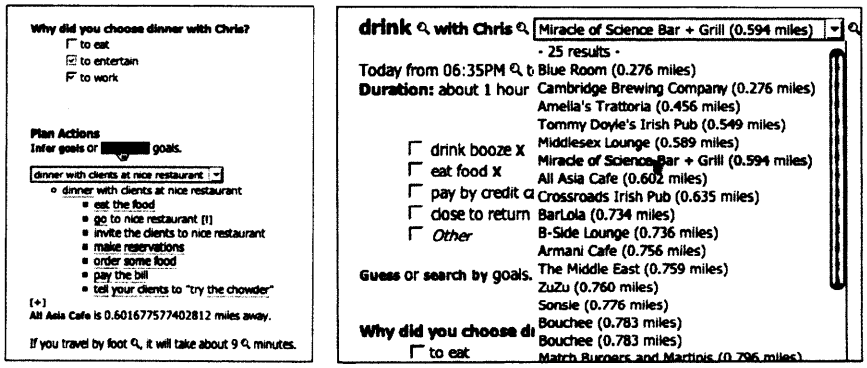


Figure 13. *EventMinder's goal-oriented approach learns from a user's decisions.*

EventMinder is a project from the MIT Media Lab that focuses on better understanding the user's desires and goals in scheduling events. To accomplish this, the project employs commonsense knowledge to interpret desires and employs a learning component to understand common event goals and user preferences *(fig. 13)*. The goal with EventMinder is largely to augment a personal calendaring assistant to understand individual user goals better, making more informed decisions, offering personalized recommendations and relevant alternatives. While Kairoscope's goals are not specifically in line with EventMinder, the ability to better understand a user, make informed decisions, and draw on common knowledge for interpretation, ultimately is. In the current prototype of Kairoscope, there is much inspiration drawn from this system, with clear future opportunities to integrate methods.

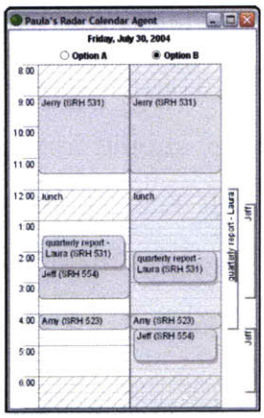


Figure 14. *RhaiCAL's option-based interface allows a user to see scheduling impacts as side-by-side options.*

RhaiCAL is another project from Carnegie Mellon looking at how humans interact with scheduling agents. In particular, it is designed around how to have a user correct mistakes, make decisions when an agent is unsure, and learn from the user's preferences over time to improve decision-making. The primary method used is to show users direct examples of the scheduling implications of both options, allowing them to compare and select one option *(fig. 14)*. Kairoscope's method of handling situations that are unresolvable follow a similar path: prompting the user about the problem, and discussing the scheduling implications of different potential actions.

PTIME is based off of a larger project at SRI on creating an assistive agent called CALO [41], particularly geared toward the workplace. PTIME uses a conversation-style interaction between users and an assistive agent, exposing the calendar for comparison and consulting in real time *(fig. 15)*.

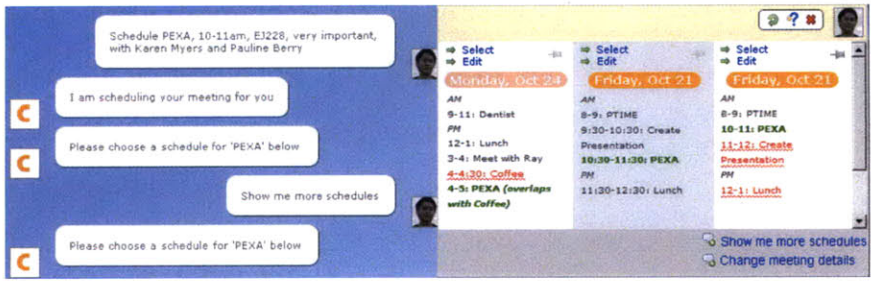


Figure 15. *PTIME's natural agent based interaction, showing side-by-side human-agent conversation and schedule.*

PTIME is interesting to look at, as it attempts to handle issues of integration, workflow, authority, and privacy. In particular, one of the challenges PTIME faced was a desire by

users to have insight and visibility into the scheduling process. This sort of response is not unique in users of assistive agents, as often a sense of lack of control can emerge should a user not understand why a decision is made. While Kairoscope is hardly immune to this issues, it largely sidesteps them by working only within user-provided constraints.

Within the context of PTIME, PLIANT, a joint project between the University of Michigan and SRI, is a learning system employed by PTIME designed to provide "adaptive assistance in an open calendaring system" [42]. The system uses a set of minimal explicit preferences, such as "I like to have long blocks of free time," augmenting them with knowledge of how the user actually interacts, often superseding the initial preferences with learned versions. Reasonable amounts of learning occurred with minimal data sets, which let PLIANT run largely unobtrusively, without having to continually query the user to make better future decisions. Kairoscope employs a much narrower learning model for user behavior, but works like PLIANT are encouraging to help do a better job of learning from a user's previous habits.

Chapter 4

Design

"Events in our lives happen in a sequence in time, but in their significance to ourselves they find their own order the continuous thread of revelation."

– Eudora Welty

There are five primary aspects of how Kairoscope looks at scheduling: improving ease of scheduling, coordinating between individuals and groups, supporting modifications that are constantly updating and evolving, using contextual data to affect schedules, and handling the precision of time

technologically. Each of these aspects has its own opportunities and requirements.

4.1 Ease of Scheduling

Scheduling is convoluted and obnoxious: a nuisance at best, impossible at worst. There have been many attempts at fixing this; the greatest successes with coordinating schedules tend to occur in specific workplaces, where meetings may be scheduled regularly in a common database. This scenario has the distinct advantage of having virtually everything scheduled: meetings, lunches, conferences, etc.

One could argue that one of the primary reasons scheduling has been less successful in the personal realm is simply that people are not dedicated enough to input every event of their lives into the calendar, often a very tedious manual procedure, with diverse systems in place to handle multi-user invitations that are compatible to varying degrees. This manual process often leads to key components of a user's day being absent from their calendars. For example, how often do users put breakfast at home into their schedules?

Therefore, one basic way to improve scheduling today would be to simply have every event that occurs in a person's life inputted into a schedule, and modified regularly. Improving input methods becomes highly relevant in this case. Kairoscope has a variety of input methods: standard GUI-based

input, text-based input, data detection in communications, and a physical input on clocks or watches. While the specific implementations of input and output will be described more in depth later in the thesis, these are the general descriptions.

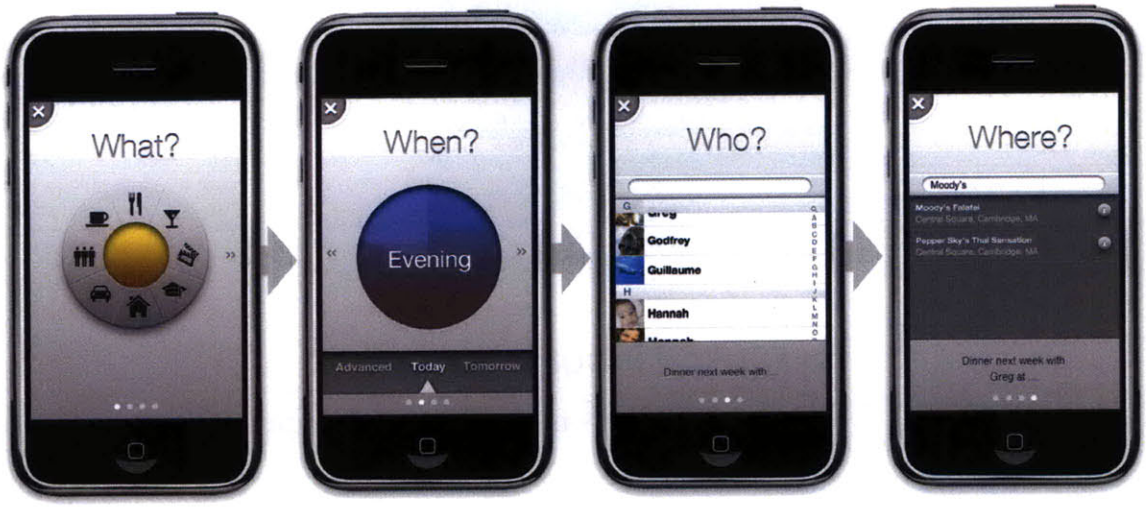


Figure 16. *Four step process to creating a new event on Kairoscope mobile application.*

Standard GUI-Based Input

Kairoscope has a mobile and desktop application to view the next upcoming event, future events, and schedule new events. Scheduling new events is a four-step process, guiding the user through choosing the event type, time and duration, location, and participants *(fig. 16)*. Any of these steps can be skipped, reducing the constraints placed on the event—requiring Kairoscope to fill in gaps.

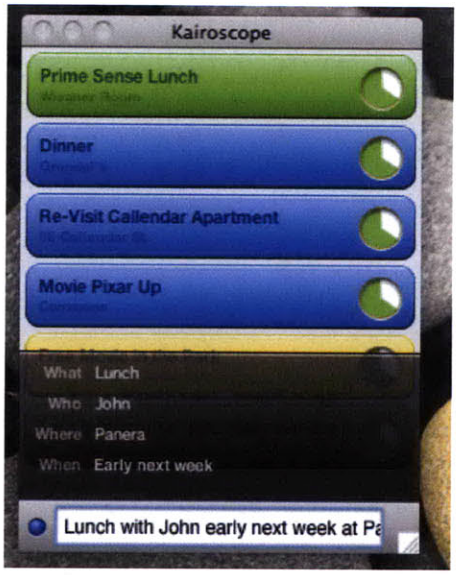


Figure 17. *Kairoscope text-based natural language input on a computer*

Text-Based Input

Kairoscope accepts natural language input as text or speech, parsing out times, locations, and participants *(fig. 17)*. This allows a user to easily input an event like this:

Lunch with John early next week at Panera

The fuzziness indicated by "early next week" gets combined with the knowledge Kairoscope has about "dinner." Any other participants indicated (e.g., John) will be contacted about the event before the event is scheduled. Any information not included or not understood will cause to Kairoscope to either fill in the gaps or to query the user (e.g., "Lunch with Tina," when Kairoscope is unaware of who Tina is, will result in a follow-up requesting information on Tina.)

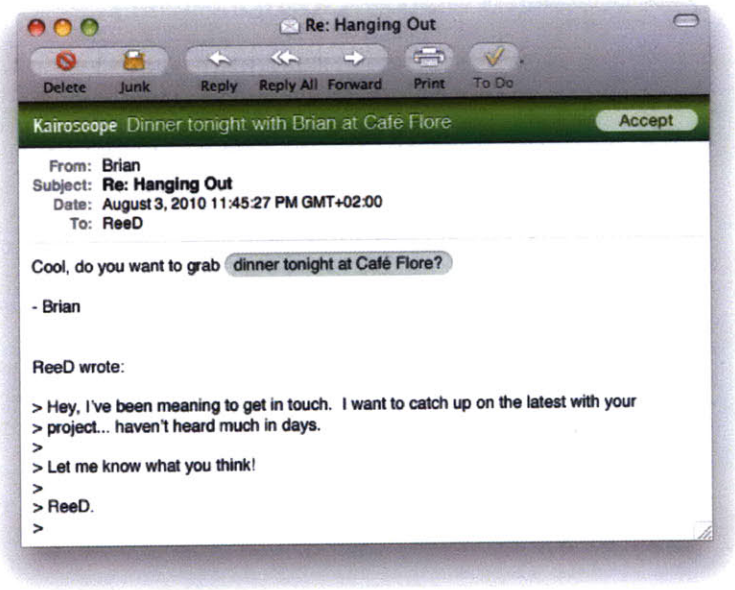


Figure 18. *Data detection in an email message.*

Data Detection in Communications

Often users receive communications, including scheduling information, in the body of the message. Kairoscope can make intelligent guesses about time, location, and participants from these messages, and offer users the ability to directly add an event to their schedule *(fig. 18)*.

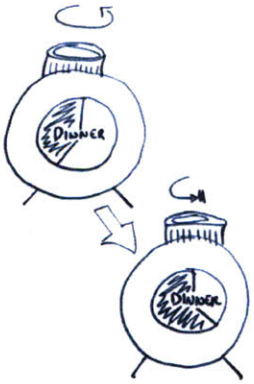


Figure 19. *Rotating the physical knob on the clock adds additional time before an event, until the user feels an end.*

Physical Inputs on Devices

Integrating scheduling into your environments, Kairoscope uses clocks both as an output device and an input device. A user can dial up a time, location, and other individuals to manage events using a physical wheel. This input has the additional advantage of offering haptic feedback to a user about modifications. A user can adjust the time of an event by rotating the wheel, but will run into a physical "end" feeling when that event is not able to be moved any further *(fig. 19)*.

Backwards Compatibility

Kairoscope can read and parse standard vCal files, and, as such, any standard calendaring application can be used to input and view schedule information. This includes Apple iCal, Microsoft Outlook, and Google Calendar. This is not intended to be used in the primary Kairoscope usage model, but is provided for negotiation and coordination with users outside of the system. All events inputted through this model are treated as immovable.

4.2 Schedule Coordination

It's generally accepted that time systems were invented primarily to allow for measurement and coordination. Coordination is especially important in a modern society, affecting everything from running a transportation system to managing a staff of employees at a store. While coordination is highly relevant, the time itself is less so: the only reason an employee may be scheduled to begin their shift at 3:00 PM is because another employee is scheduled to stop working at that same moment. The time itself is somewhat arbitrary, and generally chosen for clarity and simplicity. In the section "Time Precision," I will address how removing the specifics of timing could create an environment that maintains this simplicity without the numerical precision on the part of each individual in the system (the precision can occur on the technological side.)

Kairoscope employs an agent-based model for coordination, rather than relying on fixed times for coordination. Users are able to focus on the creation and maintenance of the events themselves, while the agents manage the optimal times and ensuring that each user is aware of when the event is occurring such that all participants show up at the same place at the same time. This model relies on agents that are able to contact each other at a moment's notice, and agents that are able to prompt a user as needed. The majority of the interaction with users is designed to occur during the creation of the event, while the agents adapt timing based on the initial parameters provided by the user.

An important note in the evaluation of Kairoscope is that it does not attempt to entirely automate scheduling between users, although it references work that has been done on this topic in the past.

4.3 On-the-Fly Modifications

Often, scheduling software falls apart when there are last-minute changes. Most standard calendaring systems are not real-time, and even systems that can notify each individual when another makes an adjustment frequently have no reliable way of ensuring that each user sees the change in time, or that the modifier is aware of everyone else's responses to the modification [43]. Kairoscope supports a multi-tiered

model where users can adjust their own timing or the timing of an event, depending on how much extra time they need.

When a user wants to push back or move forward an event, they simply indicate this to the agent, which contacts the other agent, notifies it of the adjustment, and is given an affirmative or negative response. Assuming the event is coming up shortly, the agent then may choose to notify the other user of the change, allowing them the opportunity to push back against any attempted changes *(fig 20)*. While the agents facilitate the adjustments, they do not provide a direct means for communication to discuss the modifications: the users must choose other means.

Figure 20. *From left to right: 1. John and Sarah's Coffee is approaching. 2. John realizes he's running late as coffee is approaching. 3. John rotates his Kairoscope back, which adjusts Sarah's Kairoscope in real time.*

For minor adjustments, the agents may not choose to notify the user. If the updated times fall within the same levels of scheduling probabilities, since Kairoscope does not expose time precision to each individual, adjustments can often occur without bothering the other user, or without having to even be aware of the changes. The other user may be aware of an event happening "late this afternoon," but when the agents

adjust the time from 3:13 PM to 3:25 PM due to the first user running behind on finishing a project, the impact of the change may be minimal, and thus no notice is necessary.

4.4 Intelligence

Given the desire to have a much greater data set of timing information for each person's schedule as well as the ability to adjust details rapidly on the fly, a system that looks to implement this effectively needs to use some basic contextual "intelligence" [9]. Understanding habits and regular events can provide context: when the person tends to eat meals, when the person goes to bed, when the person wakes up, when she or he has the best and worst concentration, when she or he has the most and least energy, etc.

Context is highly relevant to being able to appropriately coordinate activities. For example, to effectively schedule dinner between two individuals, it would be helpful to know when each of the individuals generally has dinner, their locations, or potentially even what kinds of food they like. This can be helpful both in the initial scheduling aspect as well as the moment itself. For example, if a user is a 40-minute drive away from the restaurant she's supposed to meet a friend at for dinner, and it's currently 30 minutes before the dinner, it's very unlikely that she will be able to make the original time, and the event timing is forced to change.

Kairoscope can collect and aggregate data over time to attempt to make better-informed decisions. In the example above, where a user wants to get dinner with a friend, Kairoscope can look back at the range of times during which each user eats dinner and when their most common and average dinner times are *(fig. 21)*, and what other events are surrounding dinner, in order to find a best-fit between the two.

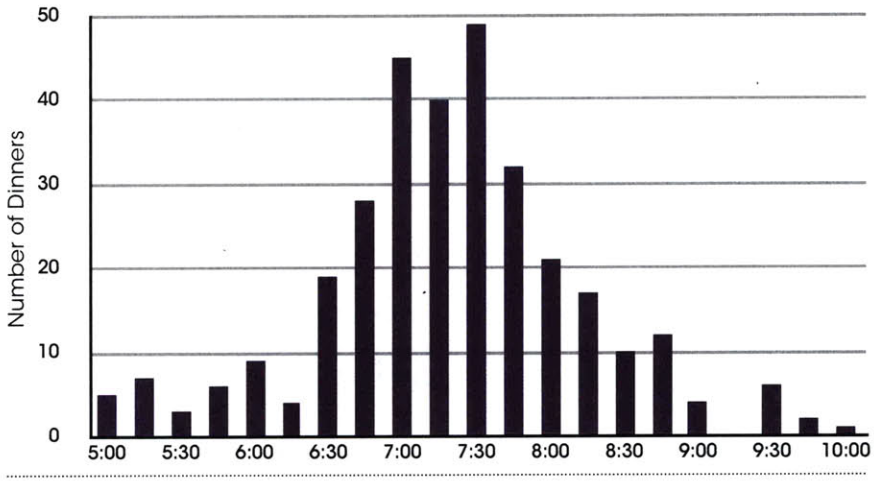


Figure 21. *Frequency of a user's dinners, organized by time of day*

This of course then needs to be able to be adjusted as each user's schedule changes—other meetings may run late, traffic may be backed up—and modify on the fly within an acceptable range of dinnertimes, or reschedule for another time if times are no longer available.

While there's a broad range of contextual knowledge that could be incorporated into this system, the primary contexts Kairoscope looks at are past timing experiences for similar events (looked at by time during a day, day of week, and

month), previous events occurring at identical locations, and previous events with the same participants.

4.5 Time Precision Handled Technologically

Perhaps the most radical approach to time in Kairoscope is the decreased emphasis on specific time-system-based times. One could begin to think of time as something "understood", rather than a precise value, a bit like temperature. While we do have measuring devices, weather widgets and television channels, and occasionally it's informative to find out what today's exact high temperature was in degrees, we're generally looking to know the answer to a few basic questions: Do I need a coat? Should I wear a t-shirt or long sleeves? Do I need an umbrella? Skirt or jeans? We have a pretty good sense of these needs simply by feel, although forecast can be helpful in giving us an indication of what conditions might exist in the near future.

If we apply this weather metaphor to time, one could imagine a system where it's relevant to get a "forecast" of events that will be occurring—for example, to see how busy a user's afternoon might be, and what she or he needs to do to prepare. Yet the precise moment at which each of those afternoon events occur may be less relevant, in the same way the difference between 76° F and 79° F is not typically relevant when one is already standing outside in a t-shirt.

Certainly we will still have a general sense of time based on the height of the sun and the relative darkness outside, but the specifics become less important. If we look at how English represents time of day, it's somewhat "fuzzy": Morning, Mid-Day, Afternoon, Evening, and Night. English uses early and late to specify a bit more precision, before falling back on the hour itself. This suggests that we tend to categorize hours within these general terms, and often in thinking about upcoming activities, we think of them as falling within one of these fuzzy moments of the day.

When scheduling precisely, we often round to a nearest hour for simplicity. We may use half-hours on occasion, or even quarter hours in specific circumstances, but rarely would we have an event for, say, 4:13 PM. This, of course, is a direct result of how our time system splits up the day, rather than any specific lack of affinity for times that fall between integer-based hours. There's no reason coffee with a friend shouldn't be scheduled from 4:13 PM to 4:47 PM, but it's a silly notion that we'd schedule such an odd time, although in reality it's no more silly than 4:00 PM to 4:30 PM. This is especially true when one considers that frequently a coffee scheduled for the latter time block ends up occurring at the former time block.

Chapter 5

Implementation

"I have to be somewhere by nine o'clock, if I hurry I can catch line 5, and leave home at quarter to nine, but the thought sounded horrible to him. This idea is based on the Calvinist view that time is valuable. The result is that nobody ever has time. Time is not precious. You are."

– Godfried Bomans

Kairoscope coordination uses peer-to-peer agents to manage and schedule each user's events in time. Each user of Kairoscope has an agent that represents the user externally to any other agent that wishes to schedule an event. The agent itself runs as a background process on each user's mobile device. In the case of the prototype, this was done on an

iPhone, but could easily be ported to run on a variety of other devices. In order to determine when events should occur, the agent is designed to use a scalable lexicon of time-based events, drawing from common knowledge until it defines a more specific set related to the individual user. Each of these agents is able to communicate in real-time with other agents to continually adapt as situations change.

5.1 Overview

The larger common knowledge information is stored in a SQL database on a server, which each agent can query when it encounters an event that it is unfamiliar with. The server responds by delivering an XML response with additional timing information if there is any available. This could be expanded to be a much more rich set of information although currently, for the purposes of the prototype, the data set is quite simple, including average time ranges for such things as meals, school and work times.

For example, if a new user schedules a dinner with another user for Thursday (day 3 of the week, where 0 is Monday), the agent may query the server to ask what range of times a dinner might occur within (1):

```
getNormalTime("dinner", "Cambridge, MA", 3);    (1)
```

The server could then respond that generally dinnertime falls between 5:35 PM and 9:42 PM on a Thursday *(fig 22)*.

```
<event>
        <type>dinner</type>
        <time dayOfWeek=3 lat=42.368945 lon=-71.109009 rad=0.5>
              <averages>
                    <start>73.315</start>
                    <end>90.471</end>
                    <max>99.201</max>
                    <min>64.185</min>
                    <mean>81.752</mean>
              </averages>
              <duration>
                    <min>2.361</min>
                    <max>8.681</max>
                    <mean>4.653</mean>
              </duration>
        </time>
</event>
```

Figure 22. *XML response from server for an event type of which the user's Kairoscope has no knowledge*

If a user requested a dinner for some time "this week," the Kairoscope client would need to query for multiple days, as the average timing for dinner may vary during the week, particularly when comparing weekends to weekdays. Since this information is culturally, regionally, and contextually variable, things like location knowledge serve to affect this information as well, and the server will respond with a closest match within a distance of the user that meets a minimum threshold of information, currently a minimum of 50 different users, with 500 different data points, although this threshold should scale. We envision this data set growing much richer as more people input this information on a regular basis and as Kairoscope aggregates the data across all of its users.

Metric Time

For the sake of easier quantifications and calculations, Kairoscope uses a version of metric time, originally conceived of in France during the French Revolution. In this implementation, Kairoscope uses a model where the day is divided into 100 parts (*centidays*). This can therefore be thought of as essentially a percentage of a day, where 00.000 is midnight at 99.999 is one second before midnight of the following day. Kairoscope is accurate to 3 decimal places, preserving a time duration accuracy of slightly greater than a current second.

In our current time system, we user 60 seconds per minute, 60 minutes per hour, and 24 hours per day. Thus, a day is made up of 86,400 seconds, the smallest generally used time unit. In metric time, there are 100 seconds per minute, 100 minutes per hour, 10 hours per day. Here, a day is composed of 100,000 seconds, each of which is just under 15% shorter than our current seconds. It seems reasonable to track times in Kairoscope to the metric second, or 1/100,000th of a day.

Timing information is stored as a decimal number with units as *centidays* (1/100th of a day), valid to three decimal places (a metric second). Unlike originally proposed metric time systems which sought 10-day weeks and 10-month years, Kairoscope continues to work under the same model of 7-day weeks and preserves the standard Gregorian calendar.

While Kairoscope does not translate between time systems, the formula for converting between our current time system and the metric time system used by Kairoscope is:

$$\frac{seconds}{864} + \frac{minutes}{14.4} + \frac{hours}{0.24} = metric\ time \qquad (Eq.\ 2)$$

So, for example, a dinner at 3:45:22 PM would be stored in Kairoscope as 65.650. An accompanying PHP script in the appendix is provided to perform this conversion.

Server Model and XML Formats

The Kairoscope server model is designed to be accessible, both as a client-API and for coordination. Scheduling an event could be tied to a specific external event timing: for example, a festival in your town can be provided timing details by the town, in the standardized kairoscope XML format:

```
<event>
        <title>Concert in the Park</title>
        <id>HE182d03wrE308hs92J</id>
        <location id="29348ehw29" />
        <type>concert</type>
        <date>2010-08-21</date>
        <url>http://mysite.com/concert</url>
        <time>
              <absolute>
                    <start>87.500</start>
                    <end>97.917</end>
              </absolute>
        </time>
</event>
```

Figure 23. *XML Format for an event to be referenced by Kairoscope.*

This XML is provided for a specific event with a beginning and end time on a specific day. This event can now be referenced by the agents when scheduling. This also allows any changes in the event's timing to be kicked back to all agents participating in the event.

In addition to the XML format, Kairoscope allows services to provide timing information through a separate tag called <kairoscope:timing> *(fig 24)*. For example, a website advertising an upcoming concert could include the following code:

```
Join us for our next concert!  We'll be playing at The Venue
<kairoscope:timing rel="http://mysite.com/concert/ks.xml"
      sid="HE182d03wrE308hs92J">at 9:30PM on Friday</kairoscope:timing>
```

Figure 24. *Kairoscope ML tag, fetches the latest timing information and interprets it in relative terms.*

This tag is interpreted by a required Javascript include (shown in Appendix B.2), fetching the latest information from the *rel* XML file, combined with any additional information from the Kairoscope server for the event with the shown *sid*. This information then replaces "at 9:30PM on Friday" with text such as "Friday night" or "Tonight after dinner." In addition, this code allows timing changes to be reflected in real time, even should this code be in an email, avoiding the necessity for constant updates and notifications. This code is ignored by browsers that do not support Javascript, and the original text will be shown instead.

One can create links to pre-existing timing content by creating a properly formatted XML file around the data to make it uniform (e.g., one could write a movies.google.com wrapper plug-in.) At this point, no data wrapper tool exists per se, although a user can also easily parse out relevant data and send the scheduling information to the Kairoscope server using the Kairoscope API (see Appendix A.)

Issues would still arise when someone says "let's see Juno on Friday" assuming there are a variety of theaters and showtimes. As such, this event may not contain enough information for Kairoscope to determine the correct showtime to choose: the user must be prompted. Alternatively, a more advanced common sense "intelligence" system could be employed to make guesses about the best time or location, although this falls outside of the scope of the current Kairoscope project. Kairoscope does not attempt to guess beyond what you've told the system, but rather it works to fill in gaps based around the constraints provided by users. The optimization of results built around these constraints could in the future be built around more intelligence.

From a scheduling perspective, when a request for an event is made from two individuals, their agents query the server for information about each other's unique address, using a dynamic DNS to ensure future communication on-demand. Each agent is able to communicate with a variety of pre-

authorized devices (e.g., physical clock devices and computers), in order to receive modifications or affect the representation. When incoming scheduling requests are received for new events, the agent confirms with the user that she or he is interested in making this plan, and then stores the event as a series of possible times.

5.2 Types of Events

There are three different broad types of events we looked at for Kairoscope: a socially coordinated activity where each user's presence is relevant to the event itself, a coordinated activity where the event will occur regardless of an individual user's presence, and an activity that the user performs alone. The first type of event is what we're most interested in from a scheduling perspective, as it poses the greatest number of challenges to current schedules, as has been discussed in this paper.

However, the second type also introduces some interesting modifications to how the interface of scheduling is handled. An example of this type of event is a class at school: a time already exists for when the class occurs, and if a user is not there on time, the class will most likely take place regardless. As a result, while pushing back an event might work for a coffee with a friend, if that event would then conflict with a more defined event in the user's schedule, there needs to be a feed-

back mechanism to let the user know they'll be missing a portion of that predefined event.

The third type of event, an activity that is performed alone, is in most ways the simplest to schedule, as it is easily modified as needed by the user. However, these events could still provide useful context to Kairoscope about how busy a particular day is for a user, what adjustments during the day might affect, and whether specific personal activities tend to have variable times or lead a user to run late. In the case of our prototype, these types of events were purely used for additional context for the schedule.

5.3 Time Fuzziness

There are a few different scenarios for the fuzziness of an event's timing that we looked at for Kairoscope:

1. Time and event are defined.
   "Let's get dinner tomorrow night."
2. Time is constrained, event is defined.
   "Let's get dinner this week."
3. Time is unknown, event is defined.
   "Let's get dinner soon."
4. Time is defined, event is unknown.
   "Let's get together tomorrow night."
5. Time is unknown, event is unknown.
   "Let's get together soon."

In the first three scenarios, while the timing of specific days varies in definition, the time of day remains known, as an event has an associated time range. Kairoscope is able to assign a possibility to each of the understood times in a user's schedule before needing to decide on the best time for the event.

The last two scenarios are more difficult, as no event information is provided. These situations could also arise if the agent simply has no knowledge of a specific event, and its associated time range (e.g., "Let's knit tomorrow.") In these cases, Kairoscope blocks off much larger chunks of time, and is sometimes forced to query the user for details about time of day or schedule context (e.g., "after dinner" or "for an hour").

Event definition is based on confidence levels, which are in turn based on the number of times an event has occurred in the past with similar conditions. When an event is specified, such as "dinner," Kairoscope begins by looking for any past events that match. This process begins by determining whether the agent is aware of this event or not. If it is not, it queries the server for information about the event. If it is, it will look at past behavior to predict future possibilities, assuming there is sufficient past data to draw reasonable conclusions, defined as a "confidence threshold." Kairoscope begins by evaluating whether there are events that match all rules, a limited initial set which is defined as:

1. *Location* – events of this type that occur at the same location.
2. *People* – events of this type that occur with the same people.
3. *Day of Week* – events of this type that occur on the same day of the week.
4. *Month* – events of this type that occur in the same month, year over year.

If there is a sufficient number of events that match all four of these categories, Kairoscope will use these values to draw predictions about when this event will occur. If there are not enough, Kairoscope will look at events that match three of these rules, and draw an average based on the relative weight of each rule. In general, Kairoscope simply applies weight to rules based on order: events that occur at the same location are more relevant than events with the same people, and so forth. This architecture allows additional rules to be introduced in the future, and ranked according to relevance.

This calculation is done by looking at an array of rules (numbered sequentially), and iterating through combinations of each, weighting them according to the order specified in the rules. A new average is calculated for each time matching an event and the rules, and is accepted if the total weighted number is greater than the confidence threshold *(fig. 25a)*.

```
rules = [1, 2, 3, 4]
for(i = length(rules) ; i <= 0 ; i--) {
        r = arrayOfCombinations(rules,i)
        for(j = 0 ; j <= length(r) ; j++) {
              totalValues += r(j) * numberOfEvents(t)
        }
        average = totalValues / length(r)
        weightedAverage = average / max(r)
        if(weightedAverage > confidenceThreshold)
              exit
}
```

Figure 25a. *Code for calculating confidence with weighted factors based on rules*

If the total number of events calculated with this model is above the confidence threshold, we will then use these, otherwise we will continue the process down to 1 *(fig. 25b)*.

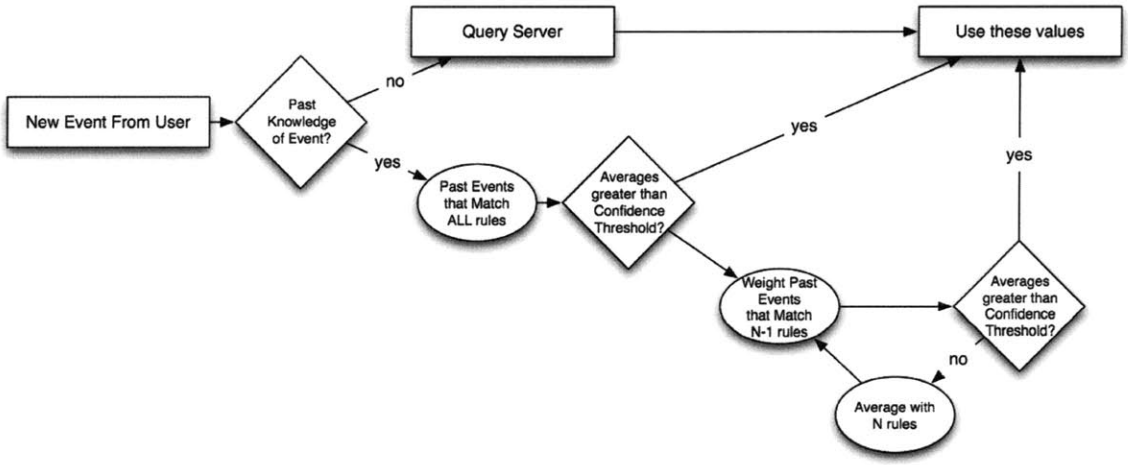


Figure 25b. *Flowchart for determining what confidence weights to use for understanding new event constraints*

Once Kairoscope has a data set to work with, it can make determinations about what times make sense given the schedule, described in the next section about "Stages."

In the scenarios where Kairoscope does not have an event definition, Kairoscope can attempt to add an event to a schedule with some default characteristics, allowing the user to adjust the event as it draws near. In scenario 4, Kairoscope looks at any events the user has scheduled for this time period, to determine probabilities for events within this time frame. For example, "tomorrow night," while "night" might indicate anywhere from 8:00 PM to 4:00 AM, a user might tend to schedule events earlier in that section. If there are not enough events in a user's history that match the time, Kairoscope will query the server for a larger definition, similar to above. In scenario 5, Kairoscope will simply look at a user's calendar, find a period of time with very few events scheduled, and place the event in the middle of this section, allowing a user to then add precision—duration, location, etc.—as they see the event approach on their schedule.

5.4 Stages of Scheduling

In addition to the fuzziness levels of an event's timing, Kairoscope moves through three different stages of scheduling for events, varying the fuzziness in each from vague to more precise, depending on their definition:

Stage 1. Potential times are noted in the schedule.

Stage 2. A tentative time has been chosen.

Stage 3. A time has been scheduled for the event.

Stage One

In stage one, Kairoscope places every possible option on the schedule, assigning different weights to each based on factors such as how busy the user is or how often the user tends to do that activity on a specific day or at a certain time. For example, if a user tends to only go out to dinner on the weekend, scheduling a dinner out next week would yield a higher weighting for dinner occurring on the weekend than during the week. This could also later be adjusted to be even more specific, based on previous patterns with this individual or interactions with varying social groups.

This method keeps all potential options open, allowing for other events that may have more constrained timings to still be scheduled. For example, if the users wanted to get dinner this week, but didn't specify a day, and another user was interested in scheduling another dinner for Thursday, it simply reduces the confidences that the dinner will occur on Thursday for the original users, limiting the number of options for when the dinner will occur between the original users.

Kairoscope accomplishes this by building a confidence table for each event, defined within the constraints given by the user for the particular event, compared to Kairoscope's knowledge of similar events in the past, adding any supplemental constraints the event specifically has, such as closing time of a restaurant when scheduling a dinner there. This initial confidence table is built based off of purely past knowl-

edge and scheduling constraints specific to the event itself, but unrelated to any other events or conflicts on the schedule, which will be factored in after the initial confidences are determined.

Figure 26 shows an example confidence table for a dinner "next week" at a restaurant that has specific hours and is closed on Mondays. All confidences outside of the green box will be dropped to 0, as they do not match one of the primary constraints.

Mon	Tue	Wed	Thu	Fri	Sat	Sun
0.007						
0.038		0.004		0.081		0.024
0.198	0.031	0.078	0.052	0.122		0.073
0.214	0.184	0.092	0.081	0.281	0.029	0.149
0.287	0.244	0.184	0.110	0.295	0.085	0.189
0.399	0.287	0.375	0.154	0.698	0.150	0.232
0.581	0.343	0.489	0.501	0.533	0.287	0.485
0.871	0.395	0.588	0.439	0.896	0.343	0.363
0.842	0.210	0.653	0.353	0.805	0.587	0.479
0.631	0.110	0.432	0.104	0.701	0.894	0.314
0.320	0.103	0.307	0.022	0.556	0.892	0.295
0.105	0.008	0.106		0.485	0.747	0.233
0.018				0.290	0.688	0.136
				0.124	0.431	0.020
				0.003	0.210	
					0.009	

Figure 26. *Sample confidence table for dinner. Times shown in green coincide with restaurant hours.*

These confidences are not probabilities based on when the event is most likely to occur—that is, all of the confidences for

any given event do not all add up to 1.0—but rather are based on the probability that any given time slot is open and desired for the event. For example, a confidence of 1.0 for lunch at noon simply indicates that Kairoscope has a 100% confidence that noon is an ideal time for lunch.

For example, to pick a time for "dinner next week," Kairoscope looks at all past dinners. The past experiences are normalized based on the chosen time range, averaged by week. Thus, a dinner scheduled for Thursday will have a 1.0 confidence for 6:30 PM, while a dinner scheduled for "this week" may have less than a 1.0 confidence for 6:30 PM on Thursday, if the user rarely schedules dinner with others on Thursdays. There are any number of ways one might be interested in parsing historical data: average dinner time on Thursdays, on Thursdays of July, on Thursdays of the second week of each month, on Thursdays of weeks where a full moon occurs. While there are guaranteed to be a variety of factors that could help make slightly more informed decisions, Kairoscope simply treats a week as a repeating sequence. Averages are calculated by storing a point for every occurrence of an event, with a time decay formula, such that older events mean less over time, based on the number of recent events.

This confidence table is then overlaid onto the schedule and compared with any overlapping confidence values for

other events, to determine adjusted confidence. The adjusted confidence is determined based on equation (4):

$$K_{adj} = K - ((K' \div 2) \times K) \qquad \textit{(Eq. 4)}$$

So, for example, if the previously mentioned lunch had a confidence of 1.0 for 12:00, but a business meeting had a confidence of 0.75 for 12:00, the adjusted confidence for lunch would be 0.625. If the confidence for lunch at 12:30 is only 0.7, but the confidence for the business meeting at the same time is 0.05, the adjusted confidence for lunch at 12:30 would be 0.6825, a higher confidence than at 12:00.

Stage Two

In stage two, Kairoscope has chosen a tentative time for the event. This stage allows Kairoscope to indicate to the user a specific date for an event, and allows the two agents to coordinate based around a specific assumed time, or more importantly, a specific date. This is done in advance, varying based on individual preferences (planner vs. spontaneous). For the purposes of the prototype, this is simply a defined number of hours in advance, which can be modified by the user, with a default stage two occurring at 30 hours in advance. This also could be variable based on the type of event or the individuals involved, although the effectiveness of having this sort of timing context was not evaluated, and likely could add a good deal of complexity—both to the scheduling model, but also to the amount of interaction from the user.

At this point, if an event needs to move around times on the same date, Kairoscope still moves the event freely within the range of possible times. However, if the event needs to change dates, a notification must be given to the user of the proposed change with an indication of the reason, and the possibility to re-evaluate. This is designed to give the user ample time to react to any changes that fall within the easily viewable future. Defining the event's relative time and date also provide a certain amount of time for the user to be able to plan for upcoming events with a high level of confidence.

Stage two is accomplished by evaluating each possible event's confidences for any given times within the possible time range, and optimizing for earlier events. The decision behind the early optimization is based on the idea that, should an event have a confidence level that is reasonably high for an early schedule, waiting for a later time could potentially add conflicts that have not yet been scheduled. Choosing an earlier time yields a higher rate of likelihood that the event will occur.

Since stage two occurs after all known adjusted confidences have been taken into consideration, optimizing for earlier rather than later appears to be a reasonable approach. Certainly, there could be a variety of reasons an event might be preferably scheduled for a later time. The confidences can be adjusted based on any additional contextual relevance neces-

sary: a user indicates a desire for lots of time for preparation (confidences start low and increase over time), certain events have close proximity and should be scheduled at similar times (confidences are increased for event times that also match location), and so forth.

Stage Three

In stage three, Kairoscope isolates the precise time when the event will occur. In our prototype, this is done at the beginning of a day for events within half a day, with the remaining day events moving to stage three as they approach, in order to provide the user a clear itinerary for events during the day. While a specific time has been chosen, as has been discussed earlier in the thesis, the time itself is not exposed to the user by default, as Kairoscope may continue to slightly modify timing as needed throughout the day up until the event itself. The purpose of the system choosing a specific time is simply to provide the user with a chronology of events that will occur, so that, for example, a user could see that the first meeting of the day will occur after lunch.

Once in stage three, Kairoscope uses basic contextual cues to determine how events might need to adjust time, as well as each user being able to move events around loosely as needed. In the first case, the previous example of time of travel being greater than the time remaining before a meeting is an automated adjustment by Kairoscope. In the second case, the user

interface of Kairoscope allows for a user to request additional time if needed (described in the subsequent section of this thesis), which gets relayed to the other participants in real time. This provides the other individuals the option to request that the delay not occur.

5.5 Agent Interactions

Integral to the functioning of Kairoscope is how agents communicate. Kairoscope agents have a simple XML based protocol used to convey event information. Kairoscope has two levels of event details: an XML that is stored locally by an agent and an XML that is sent to external agents. The locally stored XML file contains all data an agent needs to properly understand an event and timing confidences, specific to each individual. The externally referenced and transferred XML is a limited set of data relevant to the coordination of events. Specifically the major difference between the two formats is the inclusion of coordination-weighted confidences stored locally, while only a set of combined confidences are shared.

Event Initialization

When a user attempts to create a new event involving one or more other users, the agent sends out an initial event XML file to each other agent, requesting confirmation and updates regarding their confidences within the time constraints. Each agent submits a response XML file, updating with their confi-

dences for the event timing. The originating agent is tasked with managing the confidences between each agent. These confidences are calculated as described in stage two of the scheduling process, but the are then re-weighted across multiple agents. At this time, Kairoscope does not weight different agents differently (such as could be in the case of an employee and a boss), but rather leaves these interactions and scheduling details up to each user's discretion. For backward-compatibility reasons, when scheduling with an individual who is not a Kairoscope user, the Kairoscope agent treats the event as immovable at all times, initiating the event as an invitation in .ics format.

Event Modifications

One of the additional benefits of a system where precision is not exposed to the user by default is that it makes the act of changing or adjusting events more fluid. If a user sees a day is moving at a different pace than they expected, they have the opportunity to add more time before later events. When these adjustments are initiated by a user, the agent immediately contacts any other agents whose events by also be affected. This allows each agent to respond based on their needs. For example, if pushing an event forward results in a potential conflicting situation for another user, the agent will push back to the originating agent against the desired changes. This interaction can lead to two outcomes: either the originating user

decides against modifying the event or the event "pops out" of the schedule and moves into a rescheduling queue.

Events that fall into a rescheduling queue immediately look at potential times that match with the original user request, and immediately reschedule if there is a match. If there is no match within the constraints demanded by the users, the event stays in the rescheduling queue to receive specific feedback from the user about what to do. For example, if an event defined as "meeting Tuesday afternoon" gets bumped to the rescheduling queue and there are no remaining possible times on Tuesday, the user will be prompted to update the event details. The user could redefine times such as "tomorrow," "later this week," or "next week." Kairoscope will then move the event back out of the rescheduling queue and back onto the calendar. Between two agents, this process is shown on the next page *(fig. 27)*.

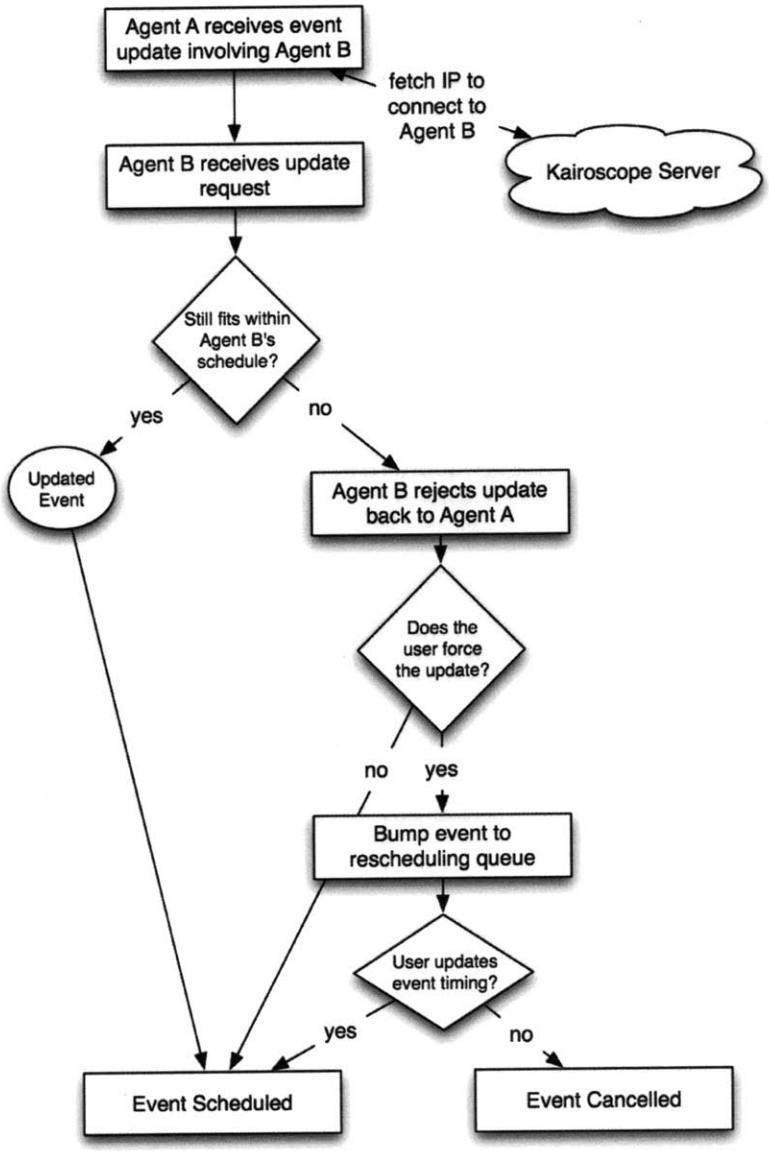


Figure 27. *Process of managing scheduling updates between two Kairoscope agents.*

If the event is in stage three of the scheduling process, this update is made apparent to the receiving user that a change is occurring. If the event is in stage two, the event can be updated without involving the user if the change does not move the event outside of a time category. For example, an event that essentially moves by 20 minutes in the afternoon will not

inform a user in stage two, but an event that moves from the afternoon to the morning will trigger a notification to the user. In stage one, the event generally can update without involving the user, assuming there is no scheduling conflict that arises. Events that change drastically (but within the constraints defined by the users) are still shown as recently updated *(fig. 28)*.

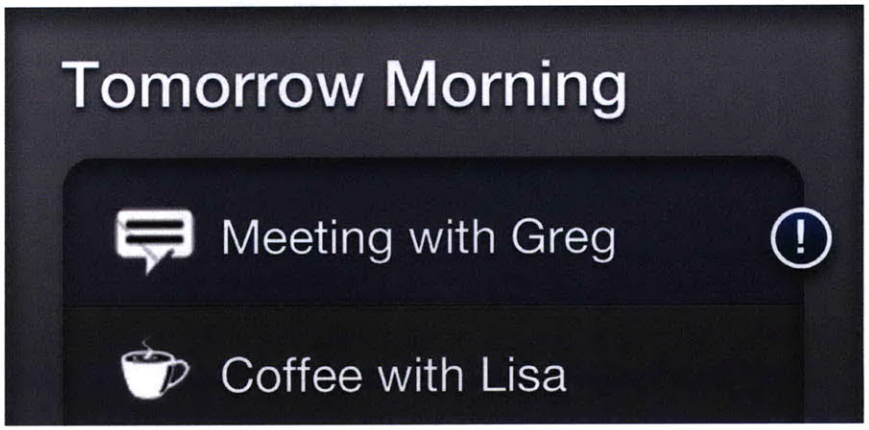


Figure 28. *An event that was recently changed is tagged in blue, without necessitating additional user notification/action.*

Real-Time Failure

While the Kairoscope system is designed around the idea of agents being able to instantly query another agent about updates, there are a variety of reasons why an agent might be unreachable by another agent. In these cases, the originating agent can "leave a message" with the Kairoscope server informing the receiving agent that it needs to contact the former. This message is composed of an updated XML event file with details of what the original agent is attempting to notify the receiving agent about. The original agent is unable to make the requested updates until hearing confirmation from

the receiving agent, and may often be forced to notify the user. The consequences of this are minimal if an event is still within stage one of scheduling process, but if the event has entered stage two, or particularly stage three, there could be specific urgent updates that must be processed quickly. In these cases, a Kairoscope agent acts as if the event is immovable to the user attempting to make an update, while providing the user a simple notification that the receiving agent is unreachable *(fig. 29)*.

Figure 29. *When one agent fails to contact another during a last-minute modification, an error is displayed to the user.*

5.6 **Location**

Location data is aggregated by monitoring the GPS location of an individual over time, sampled once per minute, stored when the location changes over a set threshold. For the purposes of the prototype, this threshold was set at ¼ of a mile.

Since it takes, on average, 3 – 7 minutes to walk ¼ of a mile, this threshold indicates a distance that Kairoscope should pay attention to, as it may impact scheduling.

Kairoscope associates GPS locations with events and locations to determine general location ranges associated with each, as well as storing time between locations. This allows for Kairoscope to begin to understand specifically times taken between distances, that may differ from initial duration predictions. For example, while Kairoscope may be able to correctly identify that the distance between a user's meeting and a café is 1 mile, there are variety of conditions that might impact the time taken to move between the two. In addition to obvious distinctions, including mode of transportation (on foot, by car, by train), a user might choose an alternative path than the most direct path, the train might have somewhat unpredictable timing, or parking might be time consuming. By performing some basic GPS tracking of users, Kairoscope can start to build a knowledge of time spent moving between two locations, gaining a sense of minimum, maximum, and average times spent.

The model used to distinguish between locations was a clustering of GPS data points, incorporated into a Markov model, as described by D. Ashbrook and T. Starner in *Learning Significant Locations and Predicting User Movement with GPS* [44]. Essentially, Kairoscope looks for time gaps or freezes in loca-

tion data that are longer than 10 minutes, shown in grey, or, more interestingly, are a similar duration to a scheduled event, shown as event bars *(fig. 30)*.

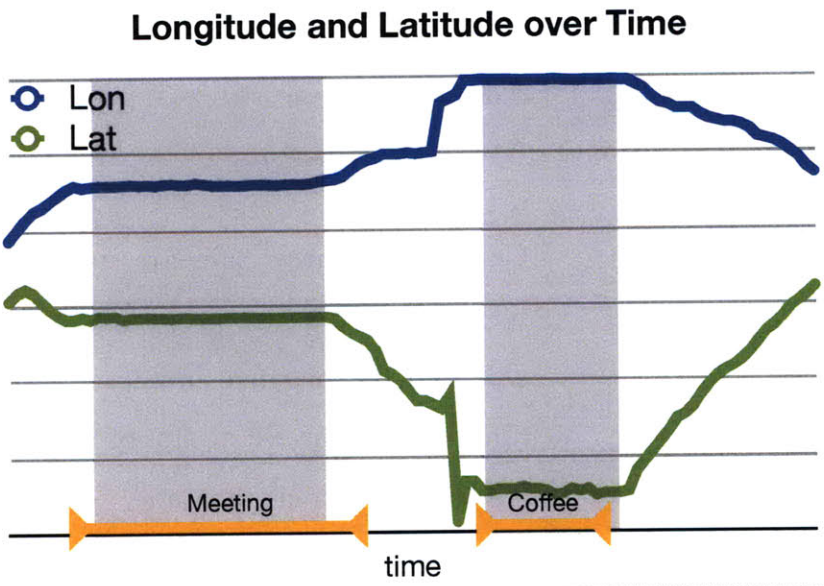


Figure 30. *Graph of longitude and latitude of a user, with event and location prediction overlays.*

Matching location data with user events can yield links between the two. Locations that do not fall within the proximate range (defined at ¼ mile) are assumed to be separate locations. If a user schedules coffee, Kairoscope may be aware of a location the user generally goes to for the duration of this event. However, there may be a variety of cafés nearby, and the same café may not consistently be chosen. Kairoscope can group proximate locations into an understood distance radius, building isolated location knowledge *(fig. 31)*.

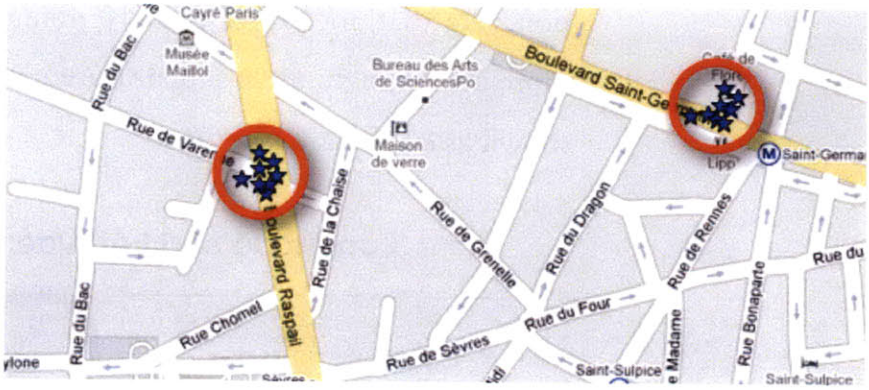


Figure 31. *Map of location points for a user during a coffee event, yielding two independent locations for the event.*

Certainly, if three cafés are situated each next to the other, Kairoscope may not properly distinguish between the three, although the negative impact is minimal, as the amount of time taken to arrive at any of the three cafés can be considered approximately the same.

When no previous location-based time data is available for events and locations, Kairoscope can query a server for averages between the two locations. Finally, if the server has no data, Kairoscope will simply make a prediction about the amount of time it takes to travel the given distance based on averages. In the Kairoscope user profile, a user indicates their preferred methods of transportation: public transport, bicycle, car, or on foot, as well as setting a range of distance chosen for each *(fig. 32)*.

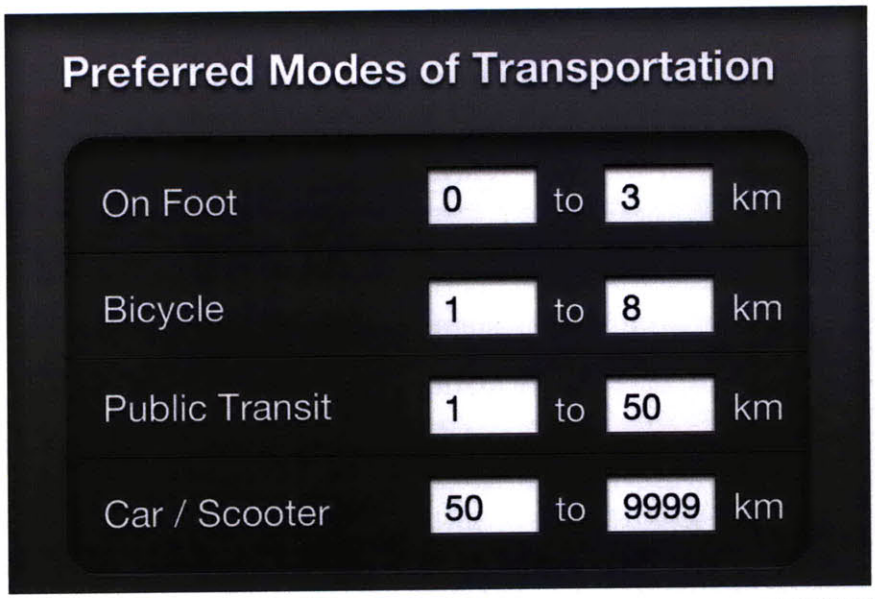


Figure 32. *Settings screen from Kairoscope allowing user to order and specify minimum and maximum distances for transportation modes.*

Kairoscope uses the following default distance ranges, based on Ewing, R. (1999) [45] :

On Foot :	0 - 1 miles
Bicycle :	0.5 - 5 miles
Public Transit :	0.5 - 60 miles
Car :	> 0.5 miles

While obviously there are a variety of factors that might affect these times (e.g., one might choose to walk or bike in good weather, but take public transit during a blizzard), Kairoscope leaves these decisions up to the individual, adjusting their time on-the-fly as needed. In certain cases, an event may be scheduled with no location information provided and no location history to draw on. In these cases Kairoscope will

simply treat the event as needing no transit time, leaving the adjustments up to the user.

5.7 User Interface

The interface for Kairoscope could be envisioned as an entire ecosystem of devices or software that can relay and input time information. For the scope of this project, a few primary interfaces were created: software on a mobile device, a personal computer, and a physical clock. Each of these devices is capable of both displaying upcoming events and adding new or modifying existing events.

Physical Clock

The clock was made by placing a screen behind existing clock hardware, to emulate the feel of a desktop clock *(fig. 33)*.

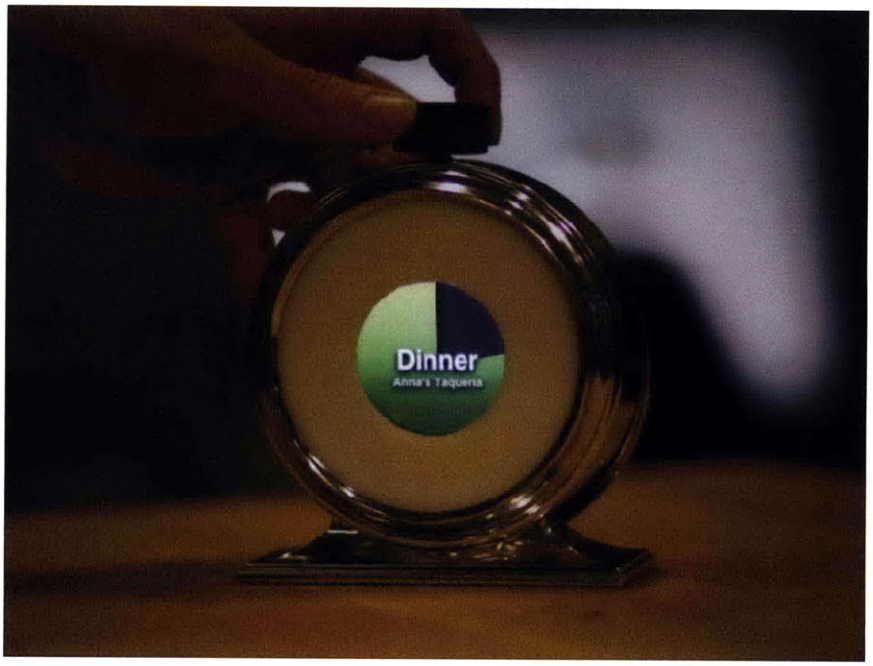


Figure 33. *Desktop Kairoscope clock, indicating an upcoming event. The haptic knob on the top allows for interaction.*

Instead of telling time, however, the clock simply displays the next upcoming event with a visual representation of the amount of time available before the next event. The visual representation would move from green to red as an event draws closer, alternatively as a pie chart style visual count-down for an event with a specific start time, or as a gradual fade for an event that occurs more loosely (for example, when the user needs to go to bed to get as much sleep as she or he normally does.) When there are no upcoming events for a while, the clock simply displays solid green.

To interact with the clock, there is a knob on top that allows a user to view upcoming events, adjust event times, and add new events. Adding an event can be triggered by clicking and holding the knob and choosing from a selection of event types and participants. Viewing upcoming events is performed by simply rotating the knob forward and backward.

Adjusting an event time occurs by pushing and rotating the knob, counter-clockwise to add more time before the event, clockwise to move the event sooner. For example, if a user sees a scheduled coffee with a friend approaching, but is in the middle of writing a paper, pushing and rotating the knob to the left will request to add some additional time before the coffee occurs. This action triggers the the agent communication described earlier, simultaneously performing this visual "rotation" on the friend's clock. The friend might then see

this change and dispute it (by rotating his or her own knob back forward again), or simply accept the adjustment.

Figure 34.
TouchSense PR-1000 knob with programmable detents, friction, and end points.

In order to discourage or prevent adjustments of certain events that might cause conflicts further down the line, a TouchSense PR-1000 knob was used *(fig. 34)*, which is equipped with programmable detents and varying levels of friction. This provides haptic feedback to the user making the timing adjustment: when user A pushes back an event that runs up against a fixed-time meeting for user B, a strong end-point is triggered, indicating that the user can not move the event past that point. Similarly, if a user has other events that occur after the one she or he is modifying that will also need re-adjusting, the friction becomes increasingly stronger the more events that need to be adjusted. In this case, one can imagine each event having a certain "weight" to it, and attempting to push multiple events rapidly becomes heavier and a bit more cumbersome. While this does not prevent the user from moving the event nonetheless, it does provide an indication as to the impacts of the modification.

The haptic feedback in the knob also becomes valuable when de-coupling an individual's schedule from an event. For example, if a meeting is going to occur regardless of a particular individual's ability to attend, if the individual is running late and attempts to push back the timing, it will have a great amount of friction to indicate the difficulty in pushing

back a meeting with a large number of participants, but at a certain point, we use a "snap" feedback, simulating a 'break.' This is also shown visually on the clock *(fig. 35)*, as the individual user's timing has now been disassociated from the meeting's timing itself. This is particularly useful in situations where a user might be forced to have conflicts with a meeting.

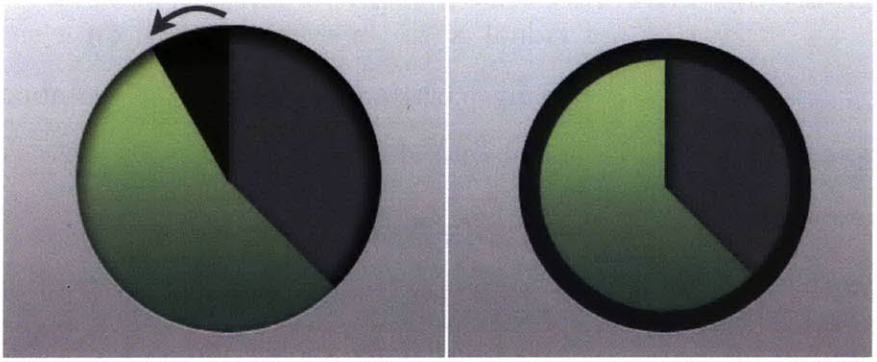


Figure 35. *Snapping behavior as an event is decoupled from an organized time event.*

This snapping behavior is similar to, yet different from, the "bumping" behavior *(fig. 36)* that occurs when a user forces an event to an irresolvable situation, causing the event to be bumped into the rescheduling queue.

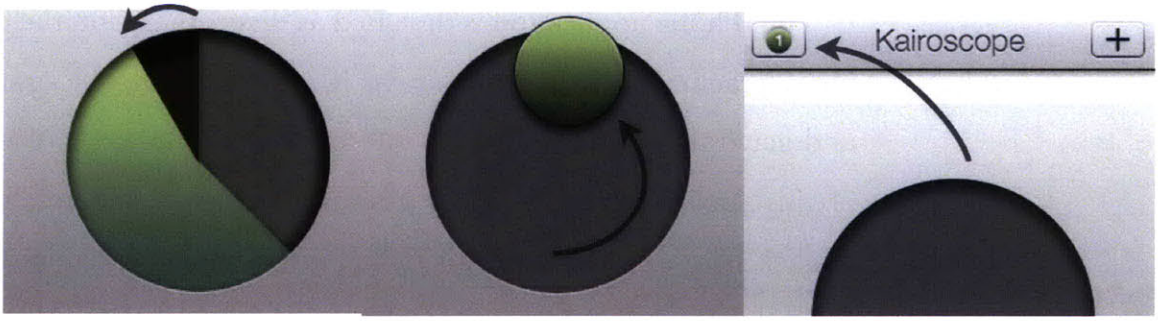


Figure 36. *Bumping behavior as an event is bumped to the rescheduling queue, indicated in the top left.*

The snapping indicates a decoupling from a coordinated event and a user's schedule, while a bumping behavior indicates a change the event itself. The former can only occur when a user is not an integral component to a meeting, such as when the meeting has many participants. As the current implementation does not provide different weights for each user, the method used to determine how important an individual is on an event is based on a simple formula of how many participants are involved, combined with the difficulty of moving the event in time.

Mobile App

In addition to the physical clock interface, Kairoscope runs as a user-facing application and a background application on the computer and on an iPhone. The method of interacting with events largely replicates the interactions described with the physical clock, albeit without the haptic feedback. The visual interface closely mimics the same snapping, bumping, and end point behaviors.

The iPhone application is designed as the primary interface for interacting with Kairoscope, as it is always with the user and generally connected to the internet. The primary screen of this application shows the upcoming event, with the ability to flick upwards to reveal all upcoming events. Upcoming events are grouped by time of day (e.g., "Afternoon", "Evening"). Each event's time is indicated as relative to the user

through a visual pie chart indicating time remaining before the event occurs, just as on the physical clock *(fig. 37)*.

Figure 37. *Kairoscope iPhone app. Left image shows next upcoming event, Right image shows all upcoming events.*

Adding a new event takes the user through some basic event-related questions, "What," "When," "Who," and "Where." Kairoscope begins by asking the user to indicate what type of event they are scheduling. In the implementation, these are grouped by most common event types and shown as a pie menu, enhancing angular memory, with the ability to see a larger list of every known event type. If there is no predetermined event type, a generic event type may be chosen, requiring specification of duration.

Figure 38. *Kairoscope iPhone app. Left image shows selection of event type, Right image shows selection of event time.*

When asking "When?" Kairoscope offers a simple radial interface for the user to rotate to indicate general time of day *(fig. 38)*. If the user has already specified a known event time that is associated with a time of day, this time of day is already specified (although the assumption may always be overridden.) This screen also offers the ability to specify larger time range, such as "Today," "Tomorrow," "This Week," "Next Week," etc. Finally, a user may specify duration by pinching in our out on the radial control, growing or shrinking the event in size from the default. For known event durations, as the control is resized, Kairoscope shows the duration

precision, but primarily indicates this using standard English terms, such as "very long dinner," or "quick dinner." If the event time is unknown, only the duration in minutes or hours is indicated.

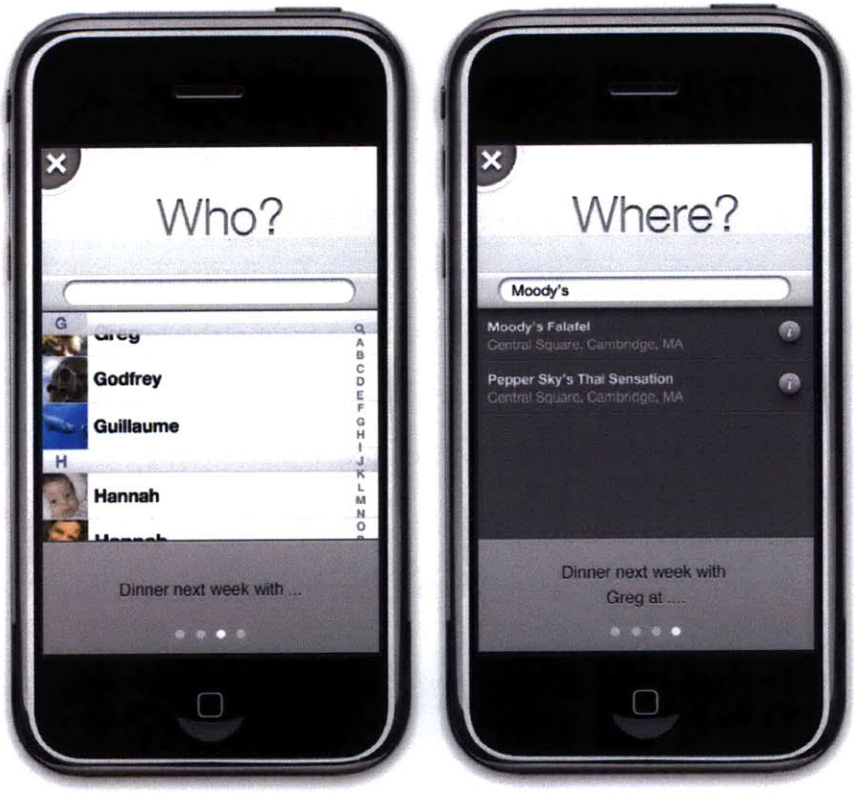


Figure 39. *Kairoscope iPhone app. Left image shows selection of people, Right image shows selection of location.*

The "Who?" screen is where a user specifies any participants to coordinate with *(fig. 39)*. The current implementation uses the Facebook API to build a known set of people to connect with. Certainly, this could be augmented in the future to include those outside of a social network, but this proved an adequate method. All contacts are displayed as a list, with

those who are physically closest listed first (within 100 yards), followed by any specified favorites, and finally an alphabetical list of all contacts.

Finally, the "Where?" screen is for specifying the precise location of the event. This is presented as a list of nearby locations that match the event type (if a match is known), followed by a list of recently scheduled locations. There is a search box to find a location by its name, and the ability to specify a previously unknown location. Kairoscope can then store this location for future use, while adding GPS data later once the user has visited the location during the event. At any point, any of these four steps can be skipped, with Kairoscope filling in any gaps, as described in the "Time Fuzziness" section.

Desktop Computer Software

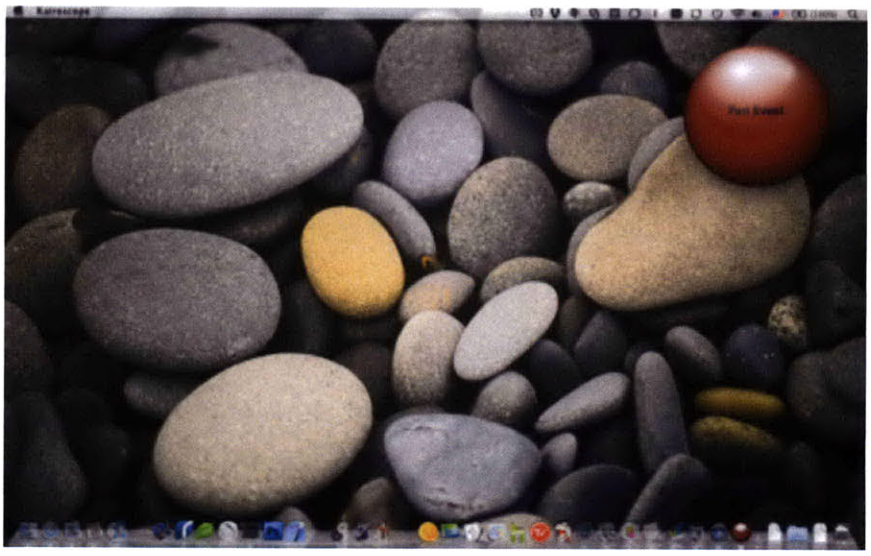


Figure 40. *Kairoscope desktop app showing time remaining before a Fun Event (none).*

The desktop computer software follows a very similar to the iPhone app model, showing alternately a floating radial clock with the next upcoming event, a menu bar clock, or an icon in the dock indicating the same *(fig. 40)*. In addition, there is a window which shows all upcoming events, and the amount of time remaining before they occur. This is accessible by clicking on any of the three Kairoscope clocks described.

While scheduling an event on the desktop has the same 4-step option as the iPhone app, one other additional interesting ability on the desktop computer is to use text- or speech-based input *(fig. 41)*. Using a slightly modified version of the Chronic natural date processing library for Ruby [46], the input is translated into usable event data, containing participants, activity, location, and time. Speech-based input is handled by MacSpeech [47] using a simple plugin to deliver the interpreted results directly to Kairoscope to parse.

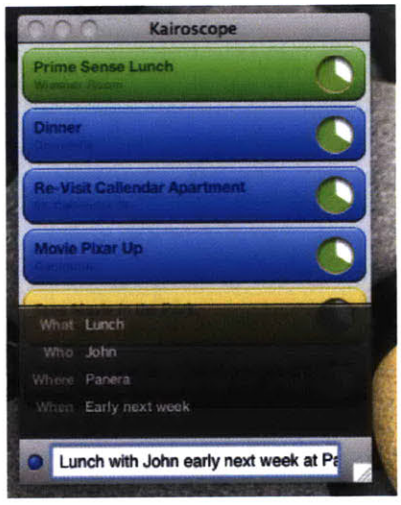


Figure 41. *Kairoscope desktop app, showing upcoming events and inputting a new event.*

Chapter 6

Evaluation

"But what minutes! Count them by sensation, and not by calendars, and each moment is a day."

– Benjamin Disraeli

In order to evaluate Kairoscope, and the concepts and design decisions that create it, limited implementations of the system were made available to a small number of users for two periods of time: a two-day period and a separate two-week period. This chapter looks at the methods used to evaluate Kairoscope, the results and responses from users, and

a discussion of the relevance. The majority of the focus of this chapter is on the user responses to Kairoscope, as limited comparison data exists for quantitative results. It is, in fact, the subjective responses that are the most interesting in line with the goals of Kairoscope described earlier.

6.1 Evaluation Design

The evaluation was structured into two sessions, a two-day period where users were assigned specific tasks to complete, and a second, two-week period where users were encouraged to use Kairoscope in their daily activities.

The users were all recruited as volunteers via an online social networking site, and were all previous users and owners of iPhone or iPod Touch devices. This criteria was selected both for simplicity of running the study, as I did not have enough devices to pass to every participant, but also to avoid learning of or inexperience with the chosen devices being a heavily contributing factor in the use of Kairoscope. The emphasis on the study was to get a sense of how users reacted to and interacted with Kairoscope as a scheduling agent that is with them at all times, rather than understanding whether the mobile device is the best choice.

Session One

In the first session, 8 users were assigned two tasks: to schedule a lunch with a "friend" and to coordinate a film

night with 4 friends. The users ranged in age from 19 to 35 years, with a median age of 25. The gender mix was event: 4 males, 4 females. 5 users were students, 2 were otherwise in academia, and 1 was a professional freelancer.

In the first task, the users were asked to simply set up a lunch with a friend any time during the week, where I played the role of the friend. Each user's calendar was pre-filled with the same set of events, such that certain times would work and others would not. The participants were told to make a best-effort attempt at scheduling the event, but conflicts happen and not every event can be successfully scheduled. The participants in this study were not in a lab setting and were not co-located with me. On day one, 4 users were tasked with scheduling the lunch with me using Kairoscope, while the other 4 users were asked to schedule however they would normally. On day two, the tasks flipped, such that the second group of 4 were performing the scheduling task in the manner of the first group.

The role that I played consisted of the following:

1. Initially suggest Thursday for lunch. (The user has a conflict.)
2. Accept the next proposed date.
3. Later, mention something came up during lunch, we have to reschedule. (The user has only 1 other lunchtime available.)

4. Accept the proposed change.

5. Separately, notify the user of a mandatory meeting that conflicts with half of the scheduled lunch.

6. Refuse the shorter lunch time request, and ask for another date. (The user will see no other options and will either choose to say the event will not happen or will reschedule another event.)

7. Accept any other proposed time / date the user suggests if they reschedule another event for the lunch.

In the second scenario, users were asked to coordinate a film night for the current week. Similarly, the group was split in two, half using Kairoscope, half scheduling in their preferred standard method. Again, the user is told to make a "best effort" attempt to schedule the event, but if it doesn't work out perfectly, it's possible they cannot get everyone to agree. The role of each of the 4 friends was again played by me, consisting of the following:

1. 3 friends will initially all agree on a time, 1 friend will disagree.

2. Upon the next proposed time, 2 friends will agree, 2 friends will disagree.

3. The third proposed time will be agreed upon by all 4 friends.

4. One friend will notify later that he is unable to attend, and either have to not attend or seek to reschedule.

5. Only 1 other time will be agreed upon by all of the friends, a time which the user has already occupied.

Each user was asked to fill out a brief survey before the study and after the study, asking them how the experience was. The success rate of each user in each scenario and with each method was noted. Users were introduced to the concept of Kairoscope and shown how the software worked for 20 minutes prior to the scenarios.

Session Two

In the second session, 13 users were chosen who had previously indicated they use a scheduling program to manage events. 16 users responded to the initial request, with 13 ultimately finishing the evaluation period. The incomplete data from the remaining 3 users was discarded. Each of the 13 users provided calendar data from the previous 4 weeks and completed a pre-survey about their scheduling habits and initial thoughts, one week prior to being introduced to Kairoscope directly. The users ranged in age from 20 to 57 years, with a median age of 32. There was a mix of college students (4 users), teachers (3 users), and company employees (6 users). The gender mix was 5 females and 8 males. The users were recruited as volunteers via an online social networking site.

In this session, users were not given specific tasks, but rather told to use Kairoscope for a two week period in their lives, and compare it to how they normally managed and thought about time. This session was designed to get initial feedback about the interface, thoughts on scheduling difficulty and stress, a better understanding of individual levels of control and feedback, and responses to social interaction optimizations. The primary points of interest lie in users' subjective responses. The surveys given to users for each of the studies are available in Appendix C.

6.1 Results

Session One

For Session One, in the first task, 7 out of the 8 participants succeeded in scheduling the lunch without using Kairoscope. Of the eight, four used only email to coordinate, two used only phone, one used both phone and email, and one used calendaring software (Google Calendar). However, of the seven, four of the users cancelled another engagement without otherwise rescheduling the event with the other person, thus indicating a preference for completing the given task over another existing event. As a result, only 4 of the 8 (50%) of the users successfully maintained all of the events in their schedule while adding the new lunch event.

As expected, all 8 users successfully completed the assigned task in Kairoscope. The survey results indicate that users found Kairoscope "surprisingly simple" and "it felt like an extension of my mind." When asked to rate their preference of using Kairoscope (1) vs. other scheduling means (5), the average score was 1.5, overwhelmingly preferring Kairoscope. One user, however, did note "I liked how easy it was, but I'm leery of seeing events moving on my calendar without my knowledge." This is the same user that currently uses Google Calendar for scheduling. When following up with the participant to clarify the concerns, as no advance calendar is ever shown, the user conceded that it was largely due to having completed the task first in the standard way, so "[the user] knew it was doing things back there somewhere!" It is also potentially worth noting that this user successfully scheduled the lunch, but ended up canceling another meeting and did not reschedule.

The user also mentioned some concerns over feeling a bit "out of control of my schedule...maybe it's not so bad, and I'm just not used to it, but I felt a bit concerned." This sort of comment is particularly of note, as this could be seen as one of the biggest challenges to the acceptance of the system. While the participants in this study did not have the opportunity to spend a long time using Kairoscope, and were unable to see the lunch event through or have actual events of consequence in their lives, it's not initially clear how the mentioned

user's concerns might become exacerbated or eased. Session two, which will be discussed later, looked a bit more at this and, while there remain concerns on varying levels of control, the general acceptance of control seemed initially positive.

Other results of note include that for the four participants who ended up canceling an event (or the lunch), each of them mentioned in their comments that they were "impressed" and "surprised" that Kairoscope was able to schedule all of the events. One user commented, "What a hassle. Makes me wonder how many things I don't do because it's such a pain to deal with, all this scheduling stuff. Ugh." Another commented "It didn't feel like I did anything. I just said...i want lunch with Reed this week, and I was done. Did I cheat?"

In the second task, 5 participants scheduled the film night without one of the friends being able to attend, 2 participants canceled another event without rescheduling to make room, and 1 participant canceled the film night altogether. The users used the same methods of scheduling as in the first task. This scenario was designed such that there was no perfect match, and even Kairoscope would be unable to find a perfect match given the constraints given. The participants would be forced to make an adjustment, regardless of the chosen time or scheduling method.

7 of the 8 users successfully completed the task using Kairoscope, with 7 moving another event to the following week,

while the remaining user moved the film night to the following week. While this outcome does result in every event being scheduled, it did not follow the initial directive to schedule the task for the same week. However, its consequences remains minimal and, in most ways, could be considered successful.

The user who had previously expressed some concern regarding levels of control commented "I appreciated that Kairoscope asked me what to do and didn't go beyond what I told [it to do]." The comments on coordinating multiple users was overwhelmingly positive, as this seems a common complaint among 6 of the participants. One of them attempted to use an online site called "Doodle" during the study to accomplish the scheduling. While initially this yielded positive results, as friends changed their availabilities it became quickly confusing and the user ultimately "spent more time doing it over email anyway."

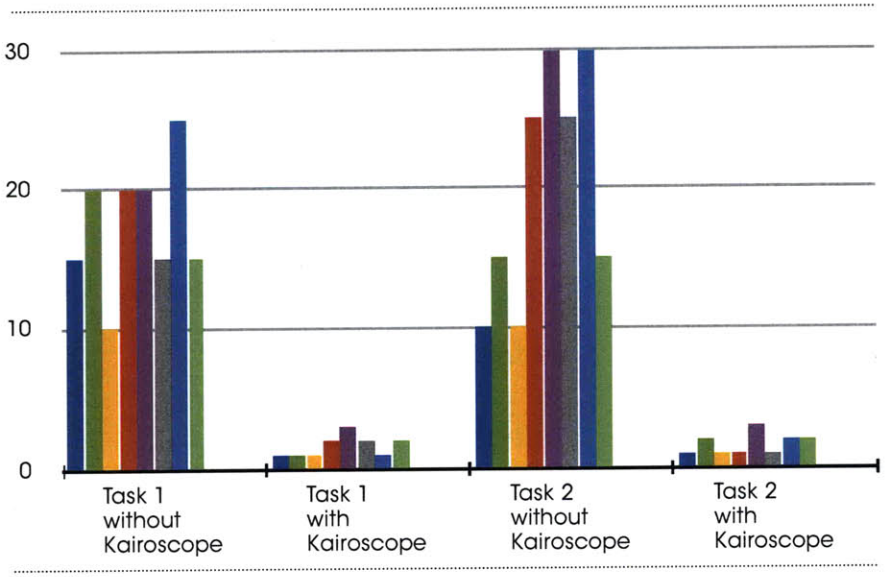


Figure 42. *Amount of time spent by user (in minutes) on scheduling two tasks, with and without Kairoscope*

The amount of time users spent interacting with Kairoscope to complete these tasks ranged from 1 minute to 3 minutes for both tasks, while the self-reported amount of time spent completing these tasks without Kairoscope ranged from 10 minutes to 25 minutes for the first task, and ten minutes to half an hour for the second *(fig. 42)*. It's important to note, however, that this is a comparison of *self-reported* amount of time for non-Kairoscope and *observed* time for Kairoscope, which may not be precise. Nonetheless, it's clear that perceived time of scheduling without Kairoscope was high, and this decreased while using Kairoscope.

Session Two

Session two results come from two surveys and statistical data from Kairoscope usage logs. One survey was done before the session (and before being introduced to kairoscope) and the other was completed after the two week period. Comparing the self-reported results from the surveys with the statistical data from Kairoscope can help gain a clearer picture of both interpretations and usage patterns. The statistical data, however, is limited to basic usage data: time spent, number of events and durations attended, and number of modifications made. In session two, the surveys and perceptions of the users are the primary interest.

Scheduling Efficiency

According to the surveys, the average time spent scheduling events went up slightly when users began using Kairoscope,

but by the second week saw a drastic decrease among all but three users. Two of these three users did in fact decrease slightly, while another two of these three users saw increases in their number of activities scheduled as well, which may have contributed to the lack of decrease in scheduling time. The average decrease in time spent scheduling per user went from around 1 ⅓ hours before Kairoscope to ⅓ of an hour in week two of the study, or around a 75% reduction in time spent scheduling *(fig. 43)*.

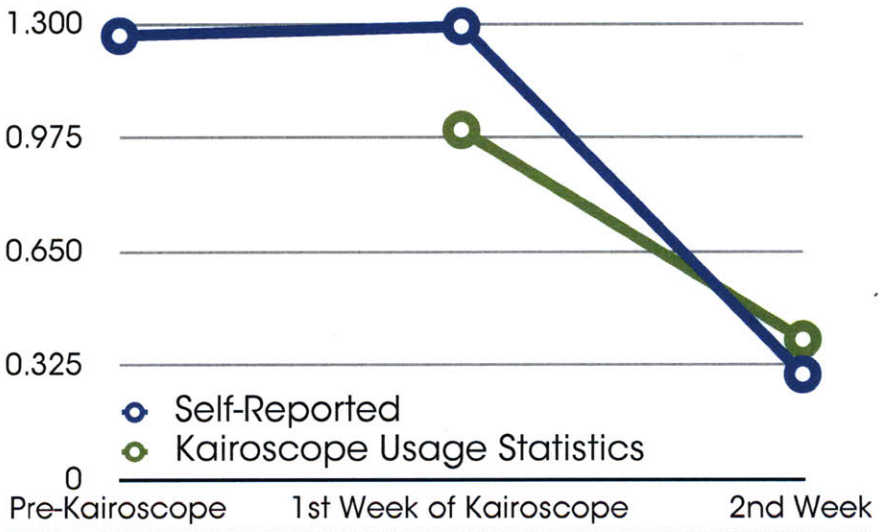


Figure 43. *Average time spent scheduling per week, before the study, during the first week, and during the second week.*

Users were generally surprisingly good at judging the amount of time they spent using Kairoscope, although they somewhat overestimated the first week and slightly underestimated the second week. While not entirely conclusive, this shift could indicate a "feeling" of simplicity among the users: the first week, when users were still figuring out details and playing

with the software, there may have been a sense that they spent more time using Kairoscope than they normally do scheduling. The second week may indicate a rapid understanding of the model, as the number of hours spent in scheduled activities did not decrease the second week, but in fact increased *(fig. 44)*.

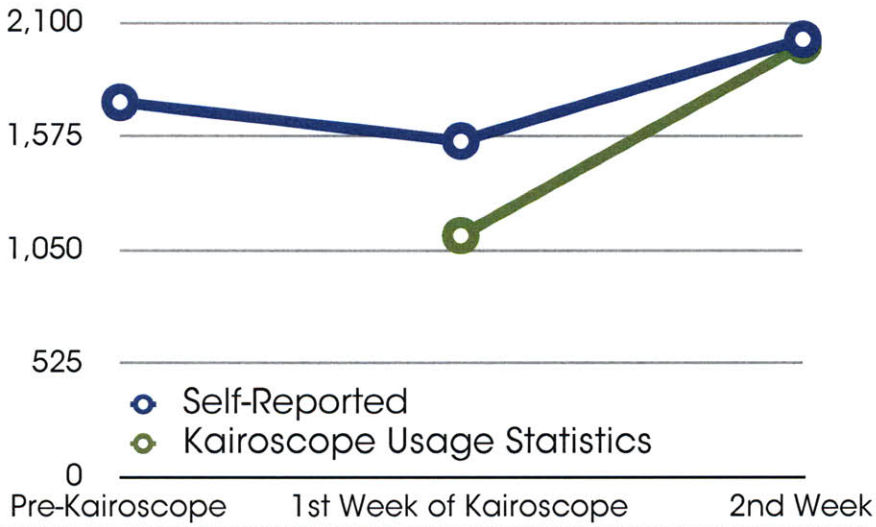


Figure 44. *Average time spent in scheduled activities per week, before the study, during the first week, and during the second week.*

It's not readily clear whether the increase in scheduled activities during the second week of the study related to the use of Kairoscope. In the user survey, users indicated a marked drop in concern over missing events after using Kairoscope: the average opinion went from *sometimes* or *often* (score of 2.38) to *rarely* to *never* (score of 0.68). In fact, every reason for missed opportunities went down on the second survey, with the largest difference found for opportunities missed due to difficulty

scheduling and the smallest for opportunities missed due to procrastination *(fig. 45)*.

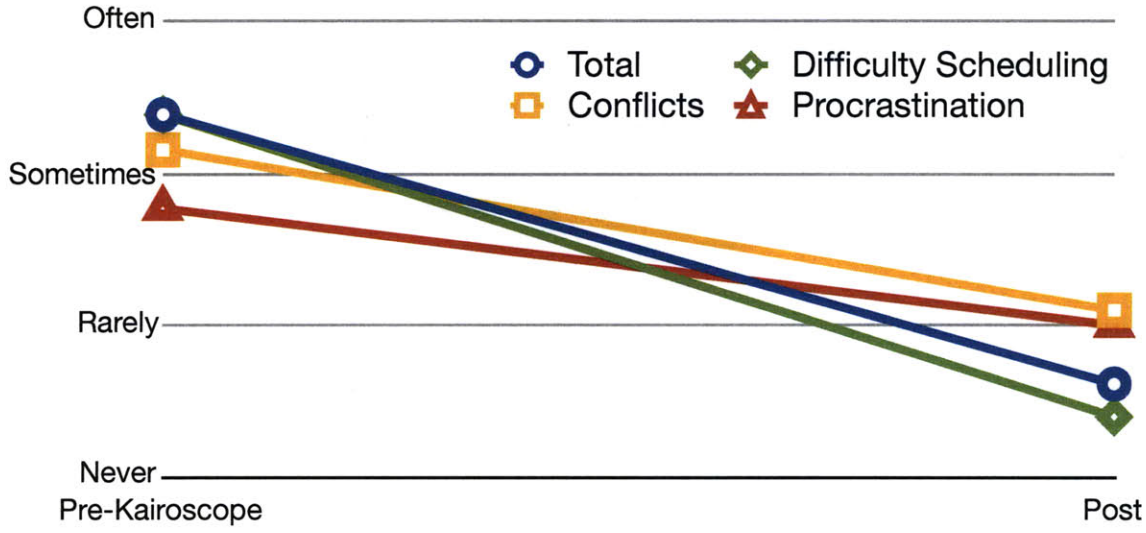


Figure 45. *Average likelihood for missed opportunities, before and after using Kairoscope, grouped by reason.*

Stress

There was a surprisingly overwhelming response to questions regarding potential stress reduction. Over ⅔ of the participants felt that Kairoscope *probably* or *definitely* let them spend less time thinking about scheduling *(fig. 47)* and less time thinking about upcoming events in general *(fig. 46)*. Astonishingly, over 90% of participants indicated that they felt Kairoscope had the potential to reduce scheduling-related stress *(fig. 48)*. Again, it's important to mention the limited sample size, yet these results are highly encouraging.

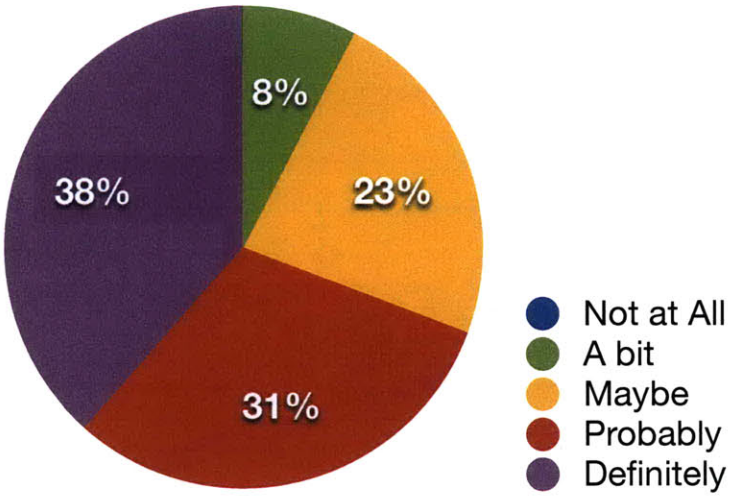


Figure 46. *Did you feel Kairoscope freed you from spending as much time thinking about upcoming events?*

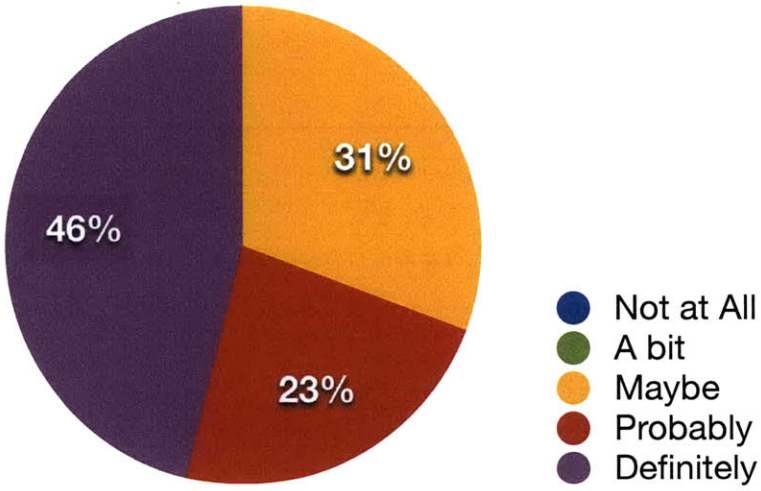


Figure 47. *Did you feel Kairoscope freed you from spending as much time thinking about scheduling?*

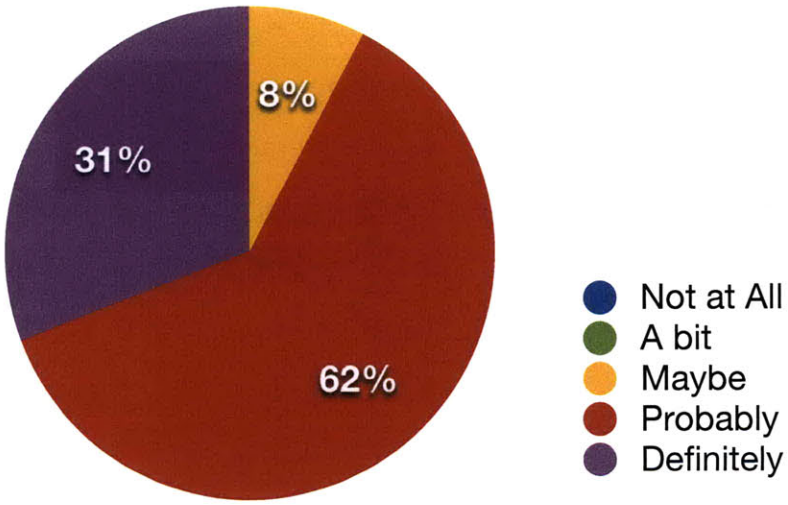


Figure 48. *Overall, do you feel Kairoscope could help reduce scheduling-related stress?*

Social Interaction

There was a modest increase in the self-reported metric of how often a user runs into friends in an unscheduled fashion (from 2.23 to 2.62 out of 6), this study was much too limited to draw any convincing conclusions on Kairoscope's ability to create more unscheduled social encounters among friends. What is encouraging, however, is that no participants felt that Kairoscope was unable to increase social interaction, and 12 out of the 13 were *somewhat* or *very interested* in increasing social opportunities for interacting with friends.

User Interface

Overall, the response to the interface was positive. The desktop computer interface was not ready for interaction at the time of the study, and the physical clock is a single prototype not suitable for distribution, so the interface questions deal only with the mobile application, the primary interface to Kairoscope for the study. Of the participants, over 50% of users "loved it", while another 31% thought it was "good." *(fig. 49)*. While the survey solicited feedback on the interface, very little additional constructive criticism or thoughts were contributed.

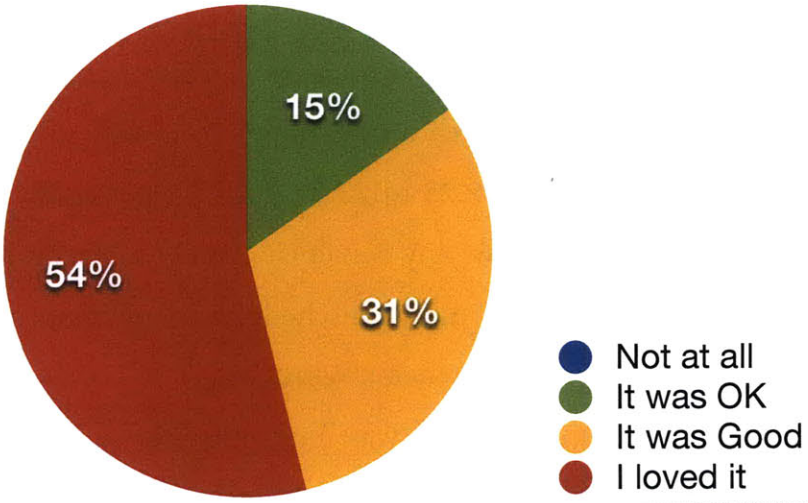


Figure 49. *How much did you like Kairoscope's interface?*

While over ⅔ of participants did not find Kairoscope's decreasing pie charts—a visual indicator of upcoming time before an event—as confusing, almost ⅓ did find it a bit confusing. This suggests that there is continued room to improve on and evaluate new methods of visualizing egocentric relative timing for upcoming events, although the pie chart seems to be a reasonable approach for many users. Less than 10% of the users found the interface "somewhat confusing," and no users marked the experience as "very confusing." *(fig. 50).*

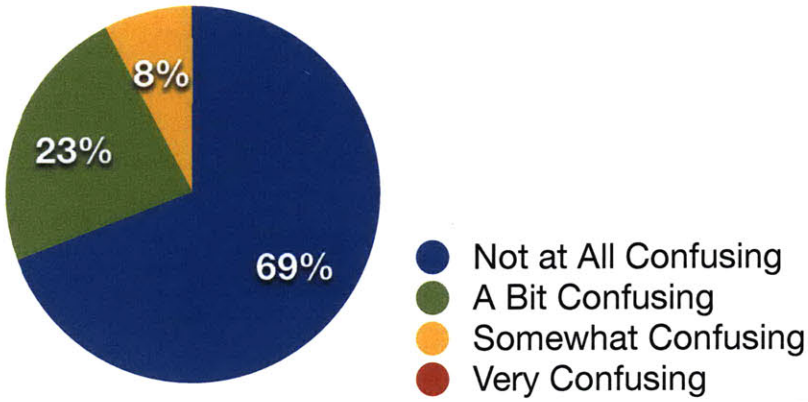


Figure 50. *How confusing was the representation of events as decreasing pie charts?*

Chapter 7

Conclusion

> *" 'Time anyhow is something made up by people,' he says.*
> *'No,' I say, 'the number of the time is something made up by people, but not the time itself. And from exactly the influence the number has on people you can derive how important all these fictions are, how they determine our life and behavior.' "*
>
> – Connie Palmen

In this thesis I have introduced Kairoscope, a novel events-based approach to interacting with, experiencing, and enjoying time. Kairoscope's egocentric approach to time navigation guides users through time from their own perspectives, relative to them, and with contextual knowledge of the

events that make up their lives. Scheduling and time management today have many challenges, and there are many attempts at working within our current models to address these. This thesis presents a different vision, questioning the premise of the human experience of time represented as a linear sequence of numbers. Through this different approach to time perception and experience, a more clear sense of time emerges, opening up new possibilities to coordinate, experience, and interact.

These possibilities are captured in the three primary goals of Kairoscope: reducing time spent managing events and stress from chronic time pressure, improving time management by more effectively organizing events, and promoting enjoyment of time through enabling opportunities for social interaction with friends. The basic premise behind Kairoscope is predicated on the notion that experiencing time without a constant exposed precision can enable these goals in ways that trying to work within the constraints of the current system cannot.

This thesis introduced these goals and the motivations behind them, looked at related and supporting work that can enhance these goals, and described how the Kairoscope system works. In addition, the methods of implementing a prototype system were described and given to users to experience

and provide a clearer picture through usage patterns and feedback.

7.1 Future Directions

Certainly a premise as bold as reevaluating our representation and perception of time has a variety of potential implications, only the most basic having been explored in this thesis. Future work can be categorized into:

1. Improving upon Kairoscope's scheduling and user understanding
2. Evaluating implications of the irrelevance of time systems in human time experiences
3. Experimenting with new ways to visualize events-based time
4. Exploring time malleability impacts, particularly on social interaction

First, Kairoscope certainly has room to grow and improve, particularly in how it handles context and learns from users over time. While Kairoscope can correctly understand past behaviors, average them based on certain characteristics, and draw reasonable conclusions in many cases, it does not understand such distinctions as a user who always gets a 10 minute coffee on days when there is a report due, a 20 minute coffee on days when it's raining, and no coffee when the user got lots of sleep. There's somewhat of a never ending supply of data

that could compensate for these situations, something a system like EventMinder could bring much additional value. There's a clear balance between trying to make too many decisions for the user and allowing the user to maintain control over the events themselves.

Second, this thesis only touches the very surface of what a society without constant awareness of precise time would be like. There are assuredly endless interesting possibilities that have not been thought of in this paper. This aspect could be evaluated from a technological perspective to determine aspects like how much definition to provide users, how peripherally-related events are represented, and optimal models for when decisions are made. This could also be evaluated from psychological, societal, or philosophical perspectives: in many ways, a consistent time system was in fact a societal "advancement." Can a society and people advance again by going back to non numerically-defined time experiences?

Third, while Kairoscope's user interface was shown to be reasonably popular by the participants of the evaluation study, it's unlikely that this project stumbled on the best possible representation of an events-based time system. It's reasonable to imagine that, given that none of the evaluation participants had experienced a similar system, that the interface corresponded to the novelty of the method. Event clouds, regular lists, event type groupings, and blurring were all visualization

ideas looked at before building the chosen prototype, and many of these had interesting potential, especially when considering that different people may have different models for events and precision.

Finally, the goal of using scheduling knowledge and time malleability to increase potential unplanned social interactions could be viewed as only one example of one of the benefits of time malleability. One could imagine events that are able to seamlessly become longer or shorter depending on each user's perception of time progression. Understanding a user's perceived elapsing of time, rather than actual elapsed time, could become the primary metric for scheduling and determining duration. Awkward social interactions that feel long could become fleeting while intense social interactions that feel "short" could extend in standard time to accommodate a longer perceived time.

It's difficult to imagine how many potential directions for future development exist; the implications are vast. Treating time as something perceived, intimate, and human rather than quantized and systematic could indeed shift navigating events back to a core human experience, freeing us from feeling like slaves to the minutes ticking away.

Maybe it's just about time.

Appendix A

Kairoscope API

Server Transactions

Timing Information

```
getNormalTime(<eventType>, <location: lon/lat | city>[, days])
updateNormalTimes(<eventType>, <location: lon/lat | city>, <date>,
                <time>, <duration>[, <locationID>])
getEventTime(<eventID>)
searchEvents(<eventName>, <location: lon/lat | city>)
```

Location Information

```
getLocationDetails(<locationID>)
searchLocations(<locationName>, <location: lon/lat | city>)
addLocation(<locationName>, <location: lon/lat>, <locationXML>)
updateLocation(<locationID>, <locationXML>)
```

Agent Transactions

```
getUpdates()
requestEvent(<eventXML>[, <additional agentIDs>])
acceptEvent(<eventID>,<updateXML>[, <additional agentIDs>])
updateEvent(<eventID>,<updateXML>[, <additional agentIDs>])
getUserDetails(<userID>)
searchUsers(<Last Name>, <First Name>)
```

getNormalTime

Gets an aggregate of timing information for a given event type and location. This can be supplemented with specific day information or specific location information to yield more precise results.

Syntax **getNormalTime(***<eventType>***,** *<location: lon/lat | city>* *[, <days>] [,<locationID>]***)**

Result Returns XML of aggregated timing information.

```
<event>
        <type>dinner</type>
        <time dayOfWeek=3 lat=42.368945 lon=-71.109009 rad=0.5>
              <averages>
                    <start>73.315</start>
                    <end>90.471</end>
                    <max>99.201</max>
                    <min>64.185</min>
                    <mean>81.752</mean>
              </averages>
              <duration>
                    <min>2.361</min>
                    <max>8.681</max>
                    <mean>4.653</mean>
              </duration>
        </time>
</event>
```

updateNormalTimes

Adds a new (anonymous) data point to the server, to be used in later aggregation for events of similar types, locations, etc.

Syntax **updateNormalTimes(***<eventType>***,** *<location:lon/lat|city>***,** *<date>***,** *<time>***,** *<duration>[,* *<locationID>]***)**

Result Returns *Success* or *Fail*

getEventTime

Retrieves information on a (publicly available) event, given an event ID.

Syntax **getEventTime(*<eventID>*)**

Result Returns *XML with all known information about the event.*

```
<event>
        <title>Concert in the Park</title>
        <id>HE182d03wrE308hs92J</id>
        <location id="29348ehw29" />
        <type>concert</type>
        <date>2010-08-21</date>
        <url>http://mysite.com/concert</url>
        <time>
              <absolute>
                    <start>87.500</start>
                    <end>97.917</end>
              </absolute>
        </time>
</event>
```

searchEvents

Retrieves the details of an event, based on event name and location. If there are multiple results, the server will return a list of possible events, details, and corresponding IDs.

Syntax **searchEvents(*<eventName>*, *<location:lon/lat|city>*)**

Result Returns *XML with details of events (including ID) found matching name and location.*

```
<results>
        <event>
              <title>Concert in the Park</title>
              <id>HE182d03wrE308hs92J</id>
              <location id="29348ehw29" />
              <type>concert</type>
              <date>2010-08-21</date>
```

```
            <url>http://mysite.com/concert</url>
            <time>
                  <absolute>
                        <start>87.500</start>
                        <end>97.917</end>
                  </absolute>
            </time>
      </event>
      <event>
            <title>Park Concert 2010</title>
            <id>TIj832fv32lpQwRF</id>
            <location id="37ei83ga" />
            <type>concert</type>
            <date>2010-08-29</date>
            <url>http://anothersite.com/concert2010/</url>
            <time>
                  <absolute>
                        <start>86.430</start>
                        <end>92.105</end>
                  </absolute>
            </time>
      </event>
</results>
```

getLocationDetails

Retrieves details of a location by ID, including name, physical location, open hours, and average timing information.

Syntax **getLocationDetails(<*locationID*>)**

Result Returns *XML with details of the location*

```
<location>
      <id>29348ehw29</id>
      <name>Zoey's</name>
      <lonlat>12.31829,-18.32892</lonlat>
      <address>38 rue Malher</address>
      <city>Paris</city>
      <state>Ile-de-France</state>
      <country>France</country>
      <zip>75004</zip>
      <phone>06 54 39 12 14</phone>
```

```
        <time>
              <averages>
                    <start>17:00:00</start>
                    <end>22:00:00</end>
                    <max>23:30:00</max>
                    <min>15:30:00</min>
                    <center>19:30:00</center>
              </averages>
              <absolutes>
                    <max>23:30:00</max>
                    <min>15:30:00</min>
              </absolutes>
        </time>
</location>
```

searchLocations

Retrieves details of a location by Name and location. If there are multiple results, the server will return a list of all possible matching locations, details, and corresponding IDs.

Syntax **searchLocations(***<locationName>, <location:lon/lat|city>***)**

Result *Returns XML with details of locations (including ID) found matching name and location.*

```
<location>
        <id>29348ehw29</id>
        <name>Zoey's</name>
        <lonlat>12.31829,-18.32892</lonlat>
        <address>38 rue Malher</address>
        <city>Paris</city>
        <state>Ile-de-France</state>
        <country>France</country>
        <zip>75004</zip>
        <phone>06 54 39 12 14</phone>
        <time>
              <averages>
                    <start>17:00:00</start>
                    <end>22:00:00</end>
                    <max>23:30:00</max>
                    <min>15:30:00</min>
```

```
                <center>19:30:00</center>
            </averages>
            <absolutes>
                <max>23:30:00</max>
                <min>15:30:00</min>
            </absolutes>
        </time>
</location>
```

addLocation

Creates a new location listing on the server from a location XML, including name and details.

Syntax **addLocation(**<*locationName*>, <*location: lon/lat*>, <*locationXML*>**)**

Result Returns newly created locationID or *fail.* If the location already exists, but the details are slightly different, the server will return a failure and the ID of the other location for the client to resolve.

```
<location>
        <id>29348ehw29</id>
</location>
```

updateLocation

Updates an existing location stored on the server with new information. For example, if a client understands the open hours of a restaurant have closed, it can notify the server.

Syntax **updateLocation(**<*locationID*>, <*locationXML*>**)**

Result Returns *Success* or *Fail.*

getUpdates

Retrieve any updates form the server about events. A client may initiate this query, for example, when it has been absent from connectivity for a while, to determine if other agents have attempted to contact it during its absence.

Syntax **getUpdates()**

Result Returns XML containing any missed event updates.

```
<results>
        <event>
              <title>Lunch with Brian</title>
              <id>E7Gwz553kt278</id>
              <location id="eh33a52c" />
              <type>lunch</type>
              <participants>
                    <id>6352829</id>
                    <id>14231</id>
              </participants>
              <duration>
                    <min>2.361</min>
                    <max>8.681</max>
                    <mean>4.653</mean>
              </duration>
              <time>
                    <day>
                          <date>2010-08-27</date>
                          <prob>0.73</prob>
                          <detail>1.025,0.37|1.030,0.34|...</detail>
                    </day>
                    <day>
                          <date>2010-08-28</date>
                          <prob>0.34</prob>
                          <detail>1.751,0.25|1.792,0.31|...</detail>
                    </day>            </time>
        </event>
</results>
```

requestEvent

Requests an event initialization to a specific agent by providing an XML of the current request. Additional agents may be specified for clarity of other negotiating agents.

Syntax **requestEvent(**<*eventXML*>*[, <additional agentIDs>]***)**

Result Returns *Success* or *Fail* depending on whether the message was received. The client must wait for an acceptEvent response to clarify whether the event is agreed upon.

acceptEvent

Informs another agent that their specified event has been accepted, and is accompanied by an updated XML file including specific scheduling updates from this agent. acceptEvent and updateEvent are virtually identical, with the exception that acceptEvent confirms or denies an initial request.

Syntax **acceptEvent(**<*eventID*>**,**<*updateXML*>
*[, <additional agentIDs>]***)**

Result Returns Success or Fail depending on whether the message was received. This does not indicate a successful schedule.

updateEvent

Sends an updated event XML after an agent has been forced to make an adjustment. acceptEvent and updateEvent are virtually identical, with the exception that acceptEvent confirms or denies an initial request.

Syntax **updateEvent(**<*eventID*>**,**<*updateXML*>
*[, <additional agentIDs>]***)**

Result Returns Success or Fail depending on whether the message was received. This does not indicate a successful schedule, and agents must both send identical updateEvent requests in sequence to confirm scheduling.

getUserDetails

Returns details of another user (and his / her agent).

Syntax **getUserDetails(*<userID>*)**

Result Returns XML of details of a user. This result will vary based on whether the user is making a public request, an already established connection request, or a certified friend request.

```
<user>
        <id>29348ehw29</id>
        <firstName>Brian</firstName>
        <lastName>Jones</lastName>
        <ip>29348ehw29.udds.kairoscope.com</ip>
        <icon>http://kairoscope.com/users/photos/29348ehw29.png</icon>
</user>
```

searchUsers

Returns all user details matching a first and last name

Syntax **searchUsers(*<last name>*, *<first name>*)**

Result Returns XML of details of every matching user. The results only include basic details, individual user details must be requested with the getUserDetails method.

```
<results>
        <user>
              <id>6352829</id>
              <firstName>Brian</firstName>
              <lastName>Jones</lastName>
              <icon>http://kairoscope.com/users/photos/6352829.png</icon>
        </user>
        <user>
              <id>14231</id>
              <firstName>Lisa</firstName>
              <lastName>Smith</lastName>
              <icon>http://kairoscope.com/users/photos/14231.png</icon>
        </user>
</results>
```

Appendix B

Scripts & Code

B.1 PHP Script to Convert Time to Metric

```
function ABTtoMT($strtime) {

        // Convert string to timestamp
        $timestamp = strtotime($strtime);

        // Parse out hours, minutes, and seconds
        $hours = date("G", $timestamp);
        $mins  = date("i", $timestamp);
        $secs  = date("s", $timestamp);

        // Convert
        $mTime = ($hours / 0.24) + ($mins / 14.4) + ($secs / 864);

        // Format to 3 decimal places
        return number_format($mTime, 3);
}
```

B.2 Javascript to fetch relative timing

```
var kairoscopeTags=document.getElementsByTagName('kairoscope:timing');

// Parse through all <kairoscope:timing> tags

for(i=0; i < kairoscopeTags.length; i++) {
        KSTag = kairoscopeTags[i];

        // Parse out all attributes
        for(j=0; j < kairoscopeTags[i].attributes.length; j++) {

                attrName = kairoscopeTags[i].attributes[j].name;
                attrValue = kairoscopeTags[i].attributes[j].value;

                // Store event SID
                if(attrName == "sid") {
                        KSsid = attrValue;
                }

                // Store event rel link
                if(attrName == "rel") {
                        KSrel = attrValue;
                }
        }

        // Query for event timing
        $.get('sidlookup.php?sid='+KSsid+'&rel='+KSrel, function(data){
                KSTag.innerHTML = data;

                KSTag.style.background = "#CCFFCC";
                KSTag.style.webkitBorderRadius = "100";
                KSTag.style.paddingLeft = "10px";
                KSTag.style.paddingRight = "10px";
        });

}
```

Appendix C

User Surveys

C.1 Group 2 Pre-Survey

1. How much time do you typically spend making plans per week? (e.g., "20 minutes")

_____

2. How many hours per week do you spend in scheduled activities? (e.g., meetings, meals, etc.)

_____

3. How often do you end up with awkward timing moments per week? (e.g., a meeting gets out early, not enough time to go anywhere.)

- ○Never
- ○1 - 2 times per week
- ○3 - 5 times per week
- ○5 - 10 times per week
- ○1 - 2 times per day
- ○3+ times per day

4. How often do you feel like you miss opportunities or not follow through due to scheduling reasons (difficulty scheduling, conflicts, procrastination)?

- ○Never
- ○Rarely
- ○Sometimes
- ○Often
- ○All the time

5. When you miss opportunities, how often is it due to... (if you never feel like you miss opportunities, skip to question #6)

	Never	Rarely	Sometimes	Often
Difficulty Scheduling	○	○	○	○
Conflicts	○	○	○	○
Procrastination	○	○	○	○

6. How often do you run into friends in an unscheduled fashion?

- ○Never or Very Rarely
- ○1 - 2 times per month
- ○1 - 2 times per week
- ○3 - 5 times per week
- ○5 - 10 times per week
- ○1 - 2 times per day
- ○3+ times per day

7. How interested are you in increasing opportunities for social interaction with your friends?

- ○ Not at all
- ○ Maybe
- ○ Somewhat Interested
- ○ Very Interested

C.2 Group 2 Post-Survey

1. How much time did you spend making plans in the first week? (e.g., "20 minutes")

_____

2. How much time did you spend making plans in the second week? (e.g., "20 minutes")

_____

3. Scheduling Efficiency

a. How often did you feel like you ended up with awkward timing moments per week?

- ○ Never
- ○ 1 - 2 times per week
- ○ 3 - 5 times per week
- ○ 5 - 10 times per week
- ○ 1 - 2 times per day
- ○ 3+ times per day

b. How often do you feel like you did (or would) miss opportunities or not follow through due to scheduling reasons (difficulty scheduling, conflicts, procrastination) with Kairoscope?

- ○ Never
- ○ Rarely
- ○ Sometimes
- ○ Often
- ○ All the time

c. Did you notice any potential conflicts that were resolved by Kairoscope?

- ○ Didn't Notice / Not Sure
- ○ None
- ○ Some
- ○ Many

d. If you missed opportunities, how often was it due to... (if you never feel like you miss opportunities, skip to question #4)

	Never	Rarely	Sometimes	Often
Difficulty Scheduling	○	○	○	○
Conflicts	○	○	○	○
Procrastination	○	○	○	○

e. Any other thoughts or ideas on scheduling?

4. Stress

a. Did you feel Kairoscope freed you from spending as much time thinking about scheduling?

- ○ Not at all
- ○ A bit
- ○ Maybe
- ○ Probably
- ○ Definitely

b. Did you feel Kairoscope freed you from spending as much time thinking about upcoming events?

- ○ Not at all
- ○ A bit
- ○ Maybe
- ○ Probably
- ○ Definitely

c. Overall, do you feel Kairoscope could help reduce scheduling-related stress?

- ○ Not at all
- ○ Unlikely
- ○ Maybe
- ○ Probably
- ○ Definitely

d. Were you able to avoid looking at clocks or being aware of the precise "time" ?

- ○ Never
- ○ Some of the time
- ○ Most of the time
- ○ Always

e. How comfortable were you navigating without knowing the time?

- ○ Not at all comfortable
- ○ Somewhat ucomfortable
- ○ Somewhat comfortable
- ○ Very comfortable

f. Are there items you'd want to have to build a better picture of your time, before using Kairoscope instead

of using a clock?

g. Did you feel Kairoscope gave you more or less control over your schedule?

- ○ Much less
- ○ Less
- ○ About the same
- ○ More
- ○ Much More

h. Did you feel Kairoscope was too aggressive in making decisions for you?

- ○ Much too aggressive
- ○ Too aggressive
- ○ About right
- ○ Not aggressive enough

i. Additional comments about stress, scheduling, and Kairoscope:

5. Social Interaction

a. How often did you run into friends in an unscheduled fashion?

- ○ Never
- ○ Once
- ○ 1 - 2 times per week
- ○ 3 - 5 times per week
- ○ 5 - 10 times per week
- ○ 1 - 2 times per day
- ○ 3+ times per day

b. Did you feel like Kairoscope could increase opportunities for social interaction?

- ○ Not at all
- ○ Some
- ○ Definitely

c. Additional comments about social interaction and Kairoscope:

6. User Interface

a. How much did you like Kairoscope's interface?

- ○ Not at all
- ○ It was OK
- ○ It was good
- ○ I loved it

b. What did you think of the interface's amount of detail?

- ○ Too detailed
- ○ Just right
- ○ Not enough detail

c. How confusing was the representation of events as decreasing pie charts?

- ○ Not At All Confusing
- ○ A Bit Confusing
- ○ Somewhat Confusing
- ○ Very Confusing

d. How often did you spend looking at the next upcoming event (1-Up) vs. all upcoming events (All)

- ○ Always 1-Up
- ○ Mostly 1-Up, glanced at All occasionally
- ○ Spent equal time with both
- ○ Mostly All, but occasionally switched to 1-Up
- ○ Always All

e. Additional comments or thoughts about the Kairoscope interface:

7. Overall, did you enjoy using Kairoscope?

○Not at all
○It was OK
○I liked it
○I loved it

8. Final Question: Any additional comments on Kairoscope, thoughts, ideas, or your experience:

Chapter 8

References

"Clocks slay time... time is dead as long as it is being clicked off by little wheels; only when the clock stops does time come to life."

– William C. Faulkner,
The Sound and the Fury

1. Miller, C. R. "Kairos in the rhetoric of science" *A rhetoric of doing: Essays on written discourse.* Southern Illinois University Press, 1992, 310-326.

2. Isocrates. *Isocrates with an English Translation in three volumes*, by George Norlin, Ph.D., LL.D. Cambridge, MA, Harvard University Press; London, William Heinemann Ltd. 1980.

3. Friedman, W. J. *About Time: Inventing the Fourth Dimension.* Cambridge, MA: MIT Press. 1990.

4. Pöppel, E. "Time Perception," in Richard Held *et al.* (eds.), *Handbook of Sensory Physiology*, Vol. VIII: Perception. Berlin: Springer-Verlag. 1978.

5. McTaggart, J. "The Unreality of Time" in *Mind: A Quarterly Review of Psychology and Philosophy*: 456-473. 1908.

6. Le Poidevan, R. "Egocentric and Objective Time," *Proceedings of the Aristotelian Society*, XCIX: 19–36. 1999.

7. Garhammer, M. "Pace of life and enjoyment of life." *Journal of Happiness Studies.* 217-256. 2002.

8. Szollos, A. "Toward a psychology of chronic time pressure." *Time and Society.* 332-350. 2009.

9. Kahneman, D., Krueger, A. B., Schkade, D. A., Schwartz, N., and Stone, A. A. "A Survey Method for Characterizing Daily Life Experience: The Day Reconstruction Method." *Science.* 1776-1780. 2004.

10. Csíkszentmihályi, M. *Flow: The Psychology of Optimal Experience.* New York. 1990.

11. Csikszentmihályi, M. "Time's Winged Chariot: Reflections on the Psychology of Time," paper presented to the Conference on "Time pressure, Work-Family Interface, and Parent-Child Relationships." Univ. of Waterloo. 2002.

12. Sigman, A. "Decline In Face-to-Face Contact Linked to Biological Changes in Humans As Social Networking Increases" *Biologist.* 56(1): 14-20. 2009.

13. Mitchell, W. J., Borroni-Bird, C., and Burns, L. D. *Reinventing the Automobile: Personal Urban Mobility for the 21st Century*. Cambridge, Mass: MIT Press. 2010.

14. Cobb, S. "Social support as a moderator of life stress." in *Toward an integrated medicine: classics from Psychosomatic medicine 1959-1979.* American Psychiatric Pub. 1995.

15. Goffman, E. *Interaction Ritual: Essays on Face-to-Face Behavior.* New York: Doubleday Anchor. 1967.

16. Uvnäs-Moberg, K. "Oxytocin may mediate the benefits of positive social interaction and emotions." *Psychoneuroendocrinology.* 819-835. 1998.

17. Heaphy, E. and Dutton, J. "Positive social interactions and the human body at work: Linking organizations and physiology." *Academy of Management Review* vol. 33, 137–162. 2008.

18. Smith, D. A. *EventMinder: A Personal Calendar Assistant That Understands Events*. Master Thesis, MIT, 2007.

19. Modi, P. J., Veloso, M., Smith, S. F., and Oh, J. "CMRadar: A Personal Assistant Agent for Calendar Management." *Proceedings of the National Conference on Artificial Intelligence.* 1020-1021. 2004.

20. Maes, P. "Agents that reduce work and information overload." *Communications of the ACM.* 1994.

21. Maes, P. and Kozierok, R. "Learning interface agents." *Proceedings of the National Conference on Artificial Intelligence.* 459-459. 1993.

22. Sycara, K and Zeng, D. "Coordination of multiple intelligent software agents." *International Journal of Cooperative Information Systems*. 181-212. 1996.

23. Jennings, N. R. and Jackson, A. J. "Agent-based meeting scheduling: A design and implementation." *IEEE Electronics Letters* (1995), 350-352.

24. Garrido, L. and Sycara, K. "Multi-Agent meeting scheduling: A design and implementation." *In Proceedings of the First International Conference on Multi-Agent Systems* (ICMAS), 1995.

25. Lin S. and Hsu, J. Y. "Learning User's Scheduling Criteria in a Personal Calendar Agent." *Proceeds of TAAI* (2000), Taipei.

26. Kim, K., Paulson Jr., B. C., Petrie Jr., C. J., and Lesser, V. R. "Compensatory negotiation for agent-based project schedule coordination." *Proceedings of the Fourth International Conference on Multiagent Systems* (2000)

27. http://office.microsoft.com/en-us/outlook/

28. http://google.com/calendar

29. http://www.peoplecube.com/products-other-meeting-maker.htm

30. http://www-01.ibm.com/software/lotus/

31. http://apple.com/ical

32. http://caldav.calconnect.org/

33. Allen, D. *Getting Things Done: The Art of Stress-Free Productivity*. Penguin Books. ISBN 0-14-200028-0. 2001.

34. Francis H. & Clement V. "Getting Things Done: The Science behind Stress-Free Productivity", *Long Range Planning 41*, no. 6: 585-605. 2008.

35. http://bargiel.home.pl/iGTD/

36. http://www.omnigroup.com/products/omnifocus/

37. http://www.trgtd.com.au/

38. http://culturedcode.com/things/

39. A.Faulringand & B.A.Myers. "Enabling rich human-agent interaction for a calendar scheduling agent." *In Proc. of CHI-05*, 2005.

40. Berry, P., Conley, K., Gervasio, M., Peintner, B., Uribe, T., Yorke-Smith, N. "Deploying a Personalized Time Management Agent" *International Conference on Autonomous Agents*. 2006.

41. W. Mark & R.Perrault. "CALO: Cognitive Assistant that Learns and Organizes." http://www.ai.sri.com/project/CALO. 2005.

42. Gervasio, M., Moffitt, M., Pollack, M., Taylor, J., and Uribe, T. "Active Preference Learning for Personalized Calendar Scheduling Assistance." In *Proceedings of the International Conference on Intelligent User Interfaces*. New York: Association for Computing Machinery. 2005.

43. Hendler et al. "Calendar Agents on the Semantic Web." *IEEE Intellegent Systems* (2002), 84-86

44. D. Ashbrook and T. Starner. *Learning Significant Locations and Predicting User Movement with GPS*. 2002.

45. Ewing, R. "Best Development Practices: A Primer." *EPA Smart Growth Network*, pp. 1-29. 1999.

46. http://chronic.rubyforge.org/

47. http://www.macspeech.com/