Surface Parking Lot Machine

A Human Factors Analysis

Dustin Bachrach
Keenan May
Hana Seok

PSYC 409
April 21, 2011
# Table of Contents

**Introduction** ................................................................................................................................. 2  
**Literature Review** ............................................................................................................................. 3  
**Defining the User** ............................................................................................................................. 5  
  - User Characteristics ......................................................................................................................... 5  
  - User Profile .................................................................................................................................. 5  
**User Requirement Analysis** ............................................................................................................. 6  
  - Operational Analysis ....................................................................................................................... 6  
  - Analysis of Similar Systems ............................................................................................................ 8  
**Preliminary Survey** ............................................................................................................................ 10  
**Design Guidance Methods** ............................................................................................................ 12  
  - Function Allocation ......................................................................................................................... 12  
  - Error Analysis ............................................................................................................................... 14  
  - Flow Analysis ............................................................................................................................... 17  
**Prototype** .................................................................................................................................... 20  
  - Initial Prototype and Testing ........................................................................................................ 20  
  - Cognitive Walkthrough ................................................................................................................ 21  
  - Usability Testing .......................................................................................................................... 27  
    - Results ....................................................................................................................................... 28  
**Conclusion** .................................................................................................................................... 29  
**References** ..................................................................................................................................... 31  
**Appendix** ....................................................................................................................................... 32
Introduction

The problem we are looking at is building a better system for surface parking lots. Typical surface parking lots are located in urban areas and are flat blocks of concrete hosting less than 50 cars. Parkers pay for their spot by placing dollar bills and coins in slot in a large box labeled with their spot number. These lots are not gated and parking attendants may not be present at all times, thus such lots hold the risk of allowing illegal parkers to leave unnoticed. We wish to design a machine that will improve this system for all users involved, the parkers and the parking lot owner. The main goals of our system are the unit to be simple and easy to use, accept multiple forms of payment, allow validations for shops and businesses around the parking lot to be used as an alternative payment method, introduce a refund feature which refunds money to parkers if they do not park for the full time they paid for, and use an electronic sensor network so the system can be aware of when cars are parked and for how long. In the following report, we present a prototype and an analysis that explains the decisions we made for our design.
Literature Review

Although parking lot design is not traditionally a highly studied topic, there has been relevant research done in related areas. Initially, we planned on applying a weight-based sensor system to monitor parked cars in our parking lot design. Through our search on the literature, we sought to find an answer to our main question; whether a weight-based sensor system was feasible/possible to build. Kumar and Siddarth (2010) described the design of a sensor system which combines infrared sensors with data from existing security cameras to create a real-time log of which spots are currently filled. This system transmits its data to a centralized receiver through radio, and the relevant information is aggregated at a receiver station and made available to a supervisor via a Graphic User Interface. This method requires a large set of IR sensors to function properly. Tang et. al (2006) describes a system in which light, temperature, and sound sensors transmit data space occupancy over a wireless network, then transmit this information to a central location. This data is instantly available online in real-time, so it would be easy for an off-site manager to keep an eye on the lot. Since this is a design project and not an engineering one, it is not necessary to go into implementation details, but the literature speaks clearly on the point that lot occupancy sensing systems are feasible, and that weight sensors, which would require installing weight sensors in each parking spot, are not the way to do it. As such, our design (if implemented) would employ a WSN (wireless sensor network) of the sort described in the aforementioned studies.

The literature details several types of similar parking systems. The closest existing relative to the sort of system we are proposing here is generally referred to as a ‘pay-and-display’ system. Rye et al (2004) assessed Edinburg pay-and-display devices, and found that about 28% of uses coincided with the average workday. If this is the number of commuters, who would be regular users, somewhere (very roughly) around 70% of traffic could be new on a given day. According to Shoup (1992), 94% of US automobile commuters do not pay for parking when they are at work. As
such, we decided to design our system to be usable by the novice rather than optimized for the frequent commuter, who is probably parking in a paid lot.

A new form of this system, called pay-by-plate, has users enter their license plate number into the metering machine. Since we have only a few kiosks for a presumably large lot, and because we wanted to minimize mnemonic ability required by the user, we decided not to go with this method. Pay-by-plate designs are relatively new, and are being currently being tested in Washington, D.C. (McPhate, 2010), along with conventional parking meters augmented with a new pay-by-phone system. We considered using a pay-by-phone system, but decided that users would not want to be forced to enter their phone number. This was later supported by user testing; users tended to ignore the option to enter their number for notification, and one explicitly said that they would not give out their phone number to a parking kiosk. In addition, this is a significant step in the interface process; based off of the user profiles we selected (several of whom were not skilled technology users), we wanted to make the interface flow as obviously as possible, even at the cost of features.

As such, when the time came to do a similar system analysis, we had already discounted the use of several broad paradigms, and were able to focus on a granular feature comparison.
Defining the User

User characteristics

We have defined the users (vehicle parkers) of our surface parking lot system as English speakers over the age of 15 with normal or corrected vision (20/40 vision or better), following the Physical Qualifications for Drivers set forth in 49 C.F.R. § 391.41 (FMCSA, 2005). The interface used for payment will be designed compatible for all users with different technical competencies.

The manager of the parking lot will be someone over the age of 16, in compliance to the Fair Labor Standards Act (DOL, 2009), which specifies the age restrictions for full-time work. The manager will receive an hour of training on how to use the parking lot paying system and when/how to contact the towing company.

User Profiles

A. Parker

<table>
<thead>
<tr>
<th>Name</th>
<th>Kyle Traveler</th>
<th>Jennis Parker</th>
<th>Jimmy Sommet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age/Gender</td>
<td>16yrs old/Male</td>
<td>53yrs old/ Female</td>
<td>37yrs old/Male</td>
</tr>
<tr>
<td>Occupation</td>
<td>Student</td>
<td>Store Manager</td>
<td>Businessmen</td>
</tr>
<tr>
<td>Driving Experience</td>
<td>New driver</td>
<td>Experienced driver</td>
<td>Experienced driver</td>
</tr>
<tr>
<td></td>
<td>Poor driving skills</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Technical Competency</td>
<td>Very Competent</td>
<td>Incompetent</td>
<td>Moderately Competent</td>
</tr>
<tr>
<td>Primary Method of Payment</td>
<td>Credit Card</td>
<td>Cash</td>
<td>Credit Card, Parking Validation</td>
</tr>
<tr>
<td>Owns a Mobile Phone</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Parking Frequency</td>
<td>Seldom</td>
<td>Every other week</td>
<td>Regular Parker</td>
</tr>
<tr>
<td>Current Problems</td>
<td>Has trouble pulling up properly at entrance gates</td>
<td>Occasionally forgets where she parked her car</td>
<td>Occasionally makes a wrong estimate of how long he will park</td>
</tr>
</tbody>
</table>

B. Manager

<table>
<thead>
<tr>
<th>Name</th>
<th>Peter Hitchcock</th>
</tr>
</thead>
<tbody>
<tr>
<td>Age/Gender</td>
<td>48yrs old/Male</td>
</tr>
<tr>
<td>Technical Competency</td>
<td>Slightly Incompetent</td>
</tr>
<tr>
<td>Current Problems</td>
<td>Works with Pay and Display system; has to patrol the entire parking lot to ensure the parkers have placed a receipt on their car dashboard. The duty can be demanding in inconvenient weathers.</td>
</tr>
</tbody>
</table>
User Requirement Analysis

Operational Analysis

A. Parker

1) User: Kyle Traveler

Context of Use: The user parks his car and goes up to the paying station to pay for his parking.

User Goal: Pay for parking

Product function: Allows the user to choose his period of stay and pay for his parking.

Possible Failure:

- The user may enter his spot number incorrectly.
- The user may incorrectly estimate his period of stay (pay more/less).
- The user may forget when his parking expires.

2) User: Jennis Parker

Context of Use: The user has parked for the 2 hours and needs extend her stay.

User goal: Add more money to parking

Product function: Allows the user to add more hours and pay for her parking.

Possible Failure:

- The user may have forgotten her spot number.
- The user may enter her spot number incorrectly.
- The user may incorrectly estimate her extra period of stay (pay more/less).
- The user may forget when her parking expires.

3) User: Jimmy Sommet

Context of Use: The user returns earlier than he had expected and wants a refund.

User goal: Receive a refund

Product Function: Gives the user a refund.

Possible Failure:

- The user may not be aware of the refund function.
- The user may enter his spot number incorrectly.
- The machine may not be able to correctly identify the user (A different user may enter the spot number and take the refund instead)
4) User: Jimmy Sommet
Context of Use: The user has paid for his parking but receives a parking validation from a nearby store.
User goal: Pay with validation and receive a refund
Product Function: Authorizes the parking validation and gives the user a refund.
Possible Failure:
- The user may not be aware of the refund function.
- The user may enter his spot number incorrectly.

B. Manager
User: Peter Hitchcock
Context of Use: The manager performing his regular parking lot duties.
User Goal: Identify a car that has not paid and contact the towing company
Product Function: Notifies the manager when a parking spot has not been paid and contacts the towing company.
Product Subordinate Function:
   i) Identify the car that had been parked for more than 30 minutes without payment.
   ii) Notify the Manager and ask for authorization to contact the towing company.
   iii) Send an automated message to the towing company.
Possible Failure:
- The system may not notify and/or falsely notify the Manager.
- The system may not send a message to the towing company when order is given.
- The manager may not notice the system sending him a notification.
Analysis of Similar Systems

We compared five different parking payment systems to observe the similarities and differences of other systems. Conventional parking meters, Pay-by-Space parking system, Pay and Display system, Rice University and Rice Village parking were evaluated. Conventional parking meters are limited in their use because they accept only coins for payment and require a parking enforcement officer to ensure the drivers had paid. Furthermore, it is possible for users to not pay for their parking if they are able to find a meter with time remaining on its display. The Pay-by-Space parking system is a fairly new system used in New Brunswick, Canada (Fredericton, 2007). The system allows its users to pay at a central paying station with both cash and credit card (but not debit card according to its webpage) by simply entering their parking space number. The Pay and Display system is a more commonly used method where the user purchases parking at a paying station and places the receipt on the dashboard of the vehicle so a parking enforcement officer can observe. These systems do not require an entrance gate unlike Rice University and Rice Village parking. Rice University allows users to pay with credit card only in certain visitor parking areas. Most parking spaces are open only for staff and students who enter the parking lot by holding their identification card to the electronic reader. Rice Village parking accepts a variety of payments, having a cashier at the exit gate.

From the information we have gathered, we observed that none of the systems were able to notify the users before their parking expired and allow an off-site manager identify when a vehicle is illegally parked. The systems that did not have entrance gates were unable to provide users a refund if they left prior to the amount of time they had paid for. Thus, we wanted to include such functions in our product.
<table>
<thead>
<tr>
<th></th>
<th>Parking Meters</th>
<th>Pay-by-Space Parking System</th>
<th>Pay and Display system</th>
<th>Rice University</th>
<th>Rice Village</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entrance boom gates</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Need of Personnel</td>
<td>Parking Enforcement Officer</td>
<td>Parking lot Manager</td>
<td>Parking Enforcement Officer</td>
<td>No</td>
<td>Cashier</td>
</tr>
<tr>
<td>Central Paying Center</td>
<td>No. one for each vehicle</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Time of Payment</td>
<td>Can add money anytime during parking</td>
<td>Right After Parking</td>
<td>Right After Parking</td>
<td>When Leaving the Parking Lot</td>
<td>When Leaving the Parking Lot</td>
</tr>
<tr>
<td>Payment Options</td>
<td>Cash (Coins)</td>
<td>Cash, Credit Card</td>
<td>Cash, Credit Card</td>
<td>Credit Card, ID card</td>
<td>Cash, Credit Card, Parking Validation (Receipt)</td>
</tr>
<tr>
<td>Notifies users when parking has expired</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Manager can identify unpaid vehicle remotely</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Refund users if they leave earlier than what they paid for</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Analysis of Similar Systems Comparison Chart
Preliminary Survey

We decided to run a survey regarding usage patterns and attitudes toward parking lots and parking systems before we began the design process. The survey had a total of 19 questions, and 14 respondents who were Rice University undergraduates. Several useful things were learned. To begin with, 42% of respondents rated the commonality of them having trouble finding their car 3 or more (on a scale of 1 to 5, 1-never/5-always). We took this into account later on when we decided to print the spot number on the receipt. Secondly, we learned that ‘credit/debit’ was by far the most desirable form of payment (86%), which we ensured to support. None of the respondents preferred tokens, which was a payment method that we initially considered for our system.

Most subjects preferred the presence of an entrance gate. However, lot security (58%) was ranked less important than cost (93%) and convenience (72%). Thus, we decided that the users would benefit more from being able to enter the parking lot from any direction that is convenient for them rather than having a gate that restricts the options of entrance.

We had also considered having ‘alert’ lights above each spot to indicate how much time was left, assuming that parkers would pass by the light often enough to notice an ‘orange’ status light. Respondents did not think they would utilize such system (58% negative response). They responded similarly for automated phone calls (43% negative response). Out of all of the alert methods presented, text alerts received the most positive response (64% positive response). Interestingly, respondents were polarized on the issue of likelihood of responding to a text alert system, notifying the users that their parking is about to expire, with respondents either being highly likely or highly unlikely. However, 9 of the respondents claimed it was ‘highly likely,’ while only 3 claimed it was ‘highly unlikely.’ Our final design reflects this, as an enter phone number for notification purposes function is included, which the user can decide rather to use it or not.
Respondents tended to park for between 1 and 4 hours (86%), indicating that these time options should be present, and displayed in some detail. No one claimed to frequently park overnight.

Respondents tended to favor a touch screen display (64%) over physical buttons (36%), which was one of the reasons we decided to use a touch screen interface. This survey would have benefited from coming a bit later in the development process; when we had a better idea of what questions to ask, but useful information was gained nonetheless.
Function Allocation

Our next step was to allocate specific functions to either the man or the machine. In this process, we first listed a number of processes that we thought were important. We then went one by one and evaluated whether each process would be better handled by the user or the machine itself.

**Function: Tracking which cars in the lot is over-time.** In traditional lots, this process is assigned to the human element. Typically, a parking lot attendant would infrequently visit the site, check the cash box, and then tally up the money to see if any of the cars currently parked did not pay. This process is very difficult and costly. First, an attendant needs to be on duty and active to find parking violators, so there is a overhead cost involved. Secondly, since the cash boxes do not timestamp payments, it can be difficult to match up specific payment with a specific car. If, however, this function was assigned to the machine, no attendant would be required at all, so the effective overhead would be minimal. Each parking lot spot would need to include weight sensors under it, which would then be connected to a central computer. Whenever a weight sensor is activated, the system knows a car is parked in a specific spot. If payment for that spot does not occur, or the car is left in the spot after time has expired, the system will immediately know that there is an over-time car and can take the appropriate action. We believe that having this function assigned to the machine results in cheaper operation and more accurate results.

**Function: Calling the tow company to move an over-time car.** Once a car is over-time, the machine will immediately know of this violation so it can take the appropriate action. Depending on the preference of the manager, the system can either immediately inform the tow company of the violation with an automated phone call, or it can alert the manager via phone or text message. The manager can then make a final decision to override the system or have it phone the tow company. For this process, we dedicate the main role to the machine but offer a human intervention option.
**Function: Finding where a specific car is parked.** There are two sub-tasks of this function. The first is to remember where a user parked their car when they come back to the lot to find it. The second is to choose a spot number when a user goes to the pay station to pay for their car. Although the pay station will print the spot number on the receipt for the user, it is ultimately the human’s responsibility to know where they parked and then find that spot in the parking lot. When a user comes to the pay station for the first time, we can actually partially assign this function to the machine though. Since every spot has weight sensors, the pay station can know what the recently parked cars that have not been paid for. Then, when a newly parked user walks to the pay station, they will not have to directly enter their spot number, but instead select their spot number from a short list of only newly parked unpaid spots. This reduces the workload for the user and allows them to recognize their spot rather than recall it. By splitting the responsibilities to both man and machine, we are getting the best experience we can.

**Function: Choosing a payment method.** There are certain tasks in our system that can only be done by either man or machine. In this case, the machine is unable to know how the user wants to pay. Therefore, we defer the choice to the human, and let them pick their payment method.

**Function: Remembering when a car is about to be over-time.** Users will sometimes incorrectly anticipate how long they want to stay at a parking lot. To account for this, users will need to return to the lot to add more time for their car. It can be difficult, though, to remember when one’s time is about to expire. We prefer the user remembering their car is about to expire both to prevent them from getting towed, but also to make sure the owner gets the appropriate amount of money. We believe that a machine can do this function much better than a human. Therefore, the machine will automatically alert users when their time is running up via text message. By relying on the machine, which can easily track time, we help prevent the aforementioned experience and improve the user experience.
**Function: Finding an empty spot.** Although our system is fully aware of the empty spots in the surface parking lot, we do not feel that the user experience would be improved by having the machine control this function. Typically, finding a spot in a surface lot is not difficult, because they will not be very full, and finding a spot is trivially obvious. Another reason to not involve the machine is that users might have a different goal in where they want to park. A machine might try to find the closest open spot from where the car entered the lot, but the user might want to find the closest spot to where they want to go to after they park. Finally, a user interface to show where open spots are would be difficult and costly. Because of all these reasons, we chose for the human to find an empty spot in the parking lot.

**Function: Calculating payment amount.** There is no way a machine can infer how long a user would like to stay at a parking lot, so we have to rely in the human element for that. However, it is much easier for a machine to calculate the price of parking based on a given amount of time rather than the user doing math in their head. Therefore, our system allows the user to enter the amount of time they wish to stay at the lot, and the pay station automatically calculates the final amount based on the rate and how long they wish to stay and includes tax. By shifting this responsibility to the machine, we easily alleviate extra work from the user and improve the experience.

**Function: Entering and exiting the surface parking lot.** The machine could possibly offer the driver directions on how to enter and exit the parking lot based on where they parked, but we feel this task is better left up to the user. Parking lot design is usually very similar, so users are used to finding the exit of a surface lot without trouble.
Error Analysis

We conducted the following error analysis based on both anticipated errors and errors we saw in usability testing. We did not have large subject numbers for testing, so we included anticipated errors in this section. A major concern we wish to avoid is when a user accidently pays for an incorrect spot. This could result from the user incorrectly remembering their spot number to a misunderstanding of the user interface. Obviously, the complications of this error are great with the user possibly being towed if they do not discover the error or much frustration as they call the management staff to undo their purchase. We took two approaches to help stop this error. First, we provide error control by only allowing payment for newly parked cars. When a customer parks his car and comes to the pay station, he is only allowed to pay for a spot that has a car parked in it (based on the WSN data), and that car must be newly parked. This way, we prevent most accidents by giving a list of options of spots a user could have parked in, which greatly reduces their ability to misremember. Secondly, we help users identify errors by always displaying the spot number on every screen of the display, so at any point through the process they can be sure they are paying for the correct spot.

The next error we anticipated is that a user does not pay. Sometimes the motivations behind this are against the law, a user trying to park without paying. At other times, a user might forget, not be aware, or not have enough time to pay. For the illegal parker, we can post signs that the parking lot is fitted with electronic parking sensors, and this knowledge that they car is more likely to be towed might sway users from consciously making this error. The only things we can do to inform the other types of users is with signage that directs them to the pay station.

Users might make a mistake by underpaying for their spot. For example, a user might think they only need an hour of parking, but they might lose track of time and realize they have gone over their time. We help users identify this problem by sending a text message to a user’s phone when
they are about to go overtime. Users, of course, have the ability to opt out of this service, or ignore the message, but they will definitely have the opportunity to be informed of the possible error.

A frequent error for parkers is to over pay for time. This is the converse of the last error, but for most parking lots, this is all profit. Our system wants to also prevent against this error. If a parker is subject to towing for being overtime, a parker should also be compensated for over paying. Our system prevents this error by giving a refund, so the user may park at the lot again free of charge.

As most drivers know, remembering where one parked can be difficult. A frequent error for parkers is to forget their spot number when they come back to the lot to find their car. We help users identify this error by giving them easily memorable spot numbers. Our spot numbers will be labeled with 1 letter and 1 digit, which means people only have to remember 2 pieces of information, much less than the limit of 7 +/- 2 (Miller 56). Also, by making each segment of information in a different domain (the set of letters and the set of digits), it is impossible for the users to misremember order. Secondly, we control against this error by printing their spot number on the receipt they receive from the machine, and any text message communication also includes their spot number. With both of these methods, the user will have means of preventing this error completely.

A more technical error is when a sensor or pay station malfunctions. We control against this by noticing when unrealistic numbers are coming out of the WSN sensors. This way, we can essentially ignore the output of a sensor and notify the owner that it needs maintenance. For a better user experience, it is more important to possibly lose money on a single malfunctioning spot than to misreport a user as not paying and towing an innocent car. Secondly, we also make the action of calling the towing company require human intervention. A manager approval (if wanted by the owner of the lot) will allow a person to override any mistakes of the system.
A user error that cannot be easily fixed is if they type in the wrong phone number. Since phone numbers are supplemental and none of their functionality is required for the payment itself, we consider this error insignificant.

After users park, they need to go to the pay station to pay for their parking. We, however, do not require them to pay instantly at the moment WSN detects the vehicle. Instead, we give them a window of time (5-10 min) to go to the pay station and pay for their parking, although the amount of money they pay for starts when the vehicle is detected in the spot. An error may arise when too many people enter the lot at the same time, and there is a bottleneck at the pay station. If users have to wait in a long line to pay, they should not be in jeopardy for not paying soon enough. To control against this error, we take into account the number of cars that recently parked in the lot and the amount of activity on the pay station interface to scale the window of time a user has to pay. We believe we can eliminate this error with an effective heuristic for window scaling.

Finally, the last error we noticed is unlikely but worth mentioning. If a user parks in a spot, which was accidentally paid for by another user, or more likely overpaid for by the previous parker, they might approach the pay station and be confused that their spot is not theirs. This is a difficult error to prevent, but a user can always enter in their spot number and the system will tell them how much time is remaining. The situation will not be much different from when hours are left in conventional parking meters. We believe that after seeing the message a user can figure out what has occurred and can react appropriately.
Flow Analysis

The following diagram shows the expected flow through the interface of the payment system. When a user approaches the payment system, we identified 4 possible goals:

- Pay for a new spot
- Add money to a spot
- Use a validation
- Get a refund for over paying

All steps begin by finding the user’s spot number. In cases when they just parked, the system will offer a shortcut by presenting their spot in a list of five newly parked cars. The users that have parked previously have the option to manually enter their spot number. After the spot has been selected correctly, both new and old parkers can select the amount of time they would like to park. When time selection has been made they are forwarded to the payment screen. The users can decide between three payment methods: cash, credit, or refund tickets. If the refund ticket the user used does not contain enough balance, the users are redirected to the choose payment method screen, allowing them to pay for the rest. Such option enables the user to pay with multiple refund tickets on the same purchase. After payment, the user is asked if they are sure. In the case that they are not, no money is charged (or they get their cash back), and they are returned to the choose payment method screen where they can use a different method or simply exit. Otherwise, their payment is processed and they receive a receipt. At the time of our original flow analysis, we did not include a phone number entering option, which would occur after payment processed, so it is not pictured in our flow diagram, but it was later added to the prototype. Refund and validation tasks are very similar where the users enter their identification number printed on their receipt or insert validation ticket and receive a refund ticket (with monetary value) and a receipt. This flow diagram gives us a high level layout of the way users will progress through our system.
Flow Analysis Diagram
Prototype

Initial Prototype and Testing

After making our broad design decisions and creating a UI flow, we decided to create a prototype. We used some proprietary prototyping software, since it allowed quick and flexible changes. However, this software had some limitations, which we ran into later in testing. In our flow analysis, we actually came up with two possible flows: one in which the user selects a spot first, then an action to perform on that spot (spot-first), and another in which the user selects an action, and chooses which spot to perform that action on. Our main goal with this early usability testing was to determine which model the users liked better. We did not have a follow-up questionnaire, and we only had a few subjects go through a couple scenarios each. We produced medium-fidelity prototypes for each of these models, with other variables the same. Generally, users were more efficient using the spot-first paradigm; this was a major factor in our eventual decision to scrap the task-first paradigm. Users also had problems with the location and size of many of the buttons, and with the unnecessarily large number of clicks required to advance screens. In addition, several users did not understand whether they should use the recent-slot buttons or the manual entry button, and stared at the home screen for some time figuring this out, which is not acceptable. We generally addressed these issues in the cognitive walkthrough, but we did produce an iteration of the prototype before the walkthrough with a few changes. Originally, we had the manual spot entry button almost as big as half of the screen, with the thinking that this would make the button more prominent. However, several users actually didn't realize that this was a button, and thought it was an entry field instead, so we sized this down. Feedback was generally positive, but there were definite hanging points that needed to be fixed. We probed these in more detail during the task analysis and cognitive walkthrough.
Cognitive Walkthrough

At this point we decided to perform a cognitive walkthrough. Based on first-round usability results, and our own conclusions about the relative complexities of the home screens of the two models, we made the decision to scrap the task-first model. In addition, it became apparent during the cognitive walkthrough of the spot-first model that the flow would be significantly more complicated in the task-first model.

The first task we ran through was one in which the user needs to pay for a spot containing a recently parked car (spot A5) with a credit card. The first thing we noticed was that the recently parked spots only said the names of spaces, not what they were for. In an interface like this, where the user needs to learn on the fly and will probably only use it once, we decided to be fairly explicit with the captioning of buttons, even if it means they take up lots of screen space. With this in mind, we changed the button labels for the ‘Recently Parked’ section so that users would be less confused as to whether they should touch these buttons or the ‘Pay for spot’ button, which requires manual entry. The new button labels read ‘Quick pay for <spot>.’ In light of feedback we received during first round usability testing that the ‘pay for spot’ button was too large to be seen as a button, we reduced its size. In addition, we added a larger ‘OR’ box to make it clear to the user that either option works. However, we concluded that the home screen could probably not be improved to the extent that we would have liked, as the user still might be confused whether the ‘Quick pay’ button or the manual entry button is correct. The fact that pressing the manual entry button by mistake is a recoverable error aside; it seemed likely that some users would still have trouble distinguishing between the options. If this system were implemented more completely, we would probably use interface cues like strobing button colors to indicate that we want the eye of most users to go to the Quick pay button. As far as the cognitive walkthrough goes, the problem here is not that the user will do the wrong thing; it is that they will not know for sure that they shouldn’t have chosen the
other option. We decided that this wasn’t a solvable problem, so we moved on with the walkthrough.

We had the insight that someone who is using a Quick pay button will not want to do anything except pay for a spot, so we had the Quick pay buttons take the user straight to the time entry screen, which we looked at next. Initially, partially due to a design decision and partially due to the limitations of the prototyping program, we had it set up so the user had to click a time, note that it was correct (presumably), then click ‘next.’ We decided that this was far from our mental models of how it should be done, and that users would likely get confused when the screen did not advance when a time was entered. As such, we eliminated the ‘next’ button, and made it so that pressing a time advances the screen. This is faster and arguably more intuitive based off of the walkthrough analysis. Since the propensity for error went up a bit, we moved the ‘current time’ indicator on the next screen to a more prominent location. This brought up a discussion of what the user would do if they wanted to go back (this should technically have been a different walkthrough path, but the discussion was fruitful so the tangent was allowed). Originally we had it so that there were ‘go back and change spot’ and ‘go back and change time’ buttons. However, it was decided that this didn’t fit with the more common language of ‘back’ and ‘start over.’ This inference was later confirmed by second-round usability testing. In addition, we decided that the user would probably not assume the system was intelligent enough to save their previous entries as a system state, as it
would have to do if there was an option to go back and change selections several steps back. That is, if they went back and changed their spot, would they expect to be taken to the time selection screen or the payment screen? It is unclear, but we decided that time gains in a few situations were not worth over-complicating the entire design with state saving and non-linearity. This was also an attempt to make the system more transparent and less mysterious, since, as we established during needs and user analyses, this is an interface designed to be functional for novices, not expedient for experts.

Next, we traced the user’s likely path to the ‘select payment’ screen. This screen looked fine, as it is relatively simple and clear. For the payment screens, we decided that the user would expect the screen to advance as soon as payment was processed, and not after confirming and touching a ‘next’ button. Our prototyping software had no way of simulating this, so we inserted dummy buttons corresponding with the inserting of appropriate payment.

After this, we looked at the screen asking whether the user would like to enter their phone number or not. At first, we framed this as ‘enter your number if you think you might want a phone number later.’ However, we decided that the user would be confused with talk of refunds right after they just paid. A user doesn’t plan to require a refund, and they won’t have it on their mind when they have just paid. As such, we decided to have user identification for refunds occur based off of a
number on the receipt. The user does not have to remember this number, or even note its presence at the time of initial payment. The new phrasing on the phone number entry request screen is “would you like to enter your cell phone number for notification purposes?” We figured that the user might have questions about what this actually meant, and whether it was safe to give out their number out to a machine, so added a “why do I need this option” button. The cell number, it is explained, is purely for notification, and does not double as an ID for refunds, the latter of which is not likely to coincide with the user’s model of what a phone number is used for. We had our imaginary user assent to giving the system their cell number, and evaluated the cell entry page next. We decided to change the keypad to be as much like a cell phone keypad as possible. In addition, we added a clearly defined ‘erase’ button, using the commonly used ‘X’ symbol. This concluded walkthrough one.

Visualization of our cell phone number entry screen (left) and the instruction screen (right)

For our second walkthrough, we went through the sequence for a hypothetical user who has just received a cell phone text notification and wants validate their spot. The user would first go to the home screen, would presumably note that their spot was not in a ‘Quick pay’ position, and press the manual entry button. We decided to that we should be consistent with our input screens (since this is what the user would expect), so we moved the ‘current spot’ indicator field to a more prominent position, and got rid of the ‘next’ button, thus setting it up so pressing a letter then a
number will advance the screen. In order to make errors recoverable, we moved the ‘current spot’ status area on the next screen to a more prominent location.

The ‘task selection’ screen, we decided, was fine. For the validation screen, we once again removed the ‘next’ button, and had it simply move on when the card had been processed. Again, this is not actually possible to do with the software, so we put a token ‘insert parking validation’ button in for testing purposes. Next, it came up that the user might want to put the validation refund on a ‘money’ card, so we added a simple screen where the user can select whether they have a ‘money’ card or not. The sequence of screens is essentially linear after that, so we decided that users would not have problems with it.

![The initial design of the Enter Validation Card screen (left) and the modified version (right)](image)

We did not perform another walkthrough, but we covered the spots that were missed by the first two in discussion. First, we decided that it should be clearer for the user whether their card is a ‘money’ card or a ‘validation’ card. Up to this point, we were still debating whether we should name the ‘money’ card ‘Refillable card’, allowing users to add balance to the card or make it a one-time usable ‘Refund Ticket’. We’ll discuss more about the final decision we made in the usability testing section. As such, we decided to color the money cards blue and the validation cards yellow, and to reflect this in the interface. The idea here was to make things simpler for the user, and take away the need for them to understand what the difference really was between the two cards. It’s easier, for example, to say ‘insert yellow parking validation card now’ than to expect the user to know what
a validation card is. This also provides a second modality for coding, which is generally a good think.

Since we decided to use paper cards for validation instead of tickets, we thought it was necessary to provide a way of clearly distinguishing the two.

In addition, we decided to add an image of an example receipt and how to find the Identification number to the ID number entry screen, which is invoked if the user requests a refund. This was done to make it easier to find the ID number, since users might not respond well to a physical description of where to find it.

This concluded our cognitive walkthrough. While we did not end up following the procedure to the letter of the law in terms of performing a strict walkthrough (we definitely did not perform a jog-through), in this case allowing the discussion to flow where it needed to, based off of inferences gained from going through the user’s thought process, was a great help. Our interface gets close to repeating itself several times, so often what was gained from analysis of one screen applied to others. In addition, the cognitive walkthrough allowed us to shave off lots of unnecessary complexity.

![Visualization of the Machine Prototype.](image)

**Visualization of the Machine Prototype.** To the right of the screen, from top to bottom: Red ‘call manager’ button, receipt printer, bill acceptor, card reader/printer, change acceptor
Usability Testing

Participants: The prototype was tested on 10 Rice University undergraduate students.

Materials: We presented the users with an online-questionnaire at the end of the usability testing to assess the usability of our system. The basic three questions were adapted from the 3-item After-Scenario Questionnaire developed by IBM for subjective usability measurement. We decided to use the questionnaire given the claim that ASQ has excellent internal consistency in measuring users’ satisfaction in scenario-based computer systems usability testing (Lewis, 1995). However, the number of questions seemed rather minimal, and we were concerned that our questions were leading the users to give only positive feedback. In order to neutralize our questions, we adapted a question from the Software Usability Measurement Inventory method, asking how much the users thought our system was ‘unnecessarily’ complex. We also added another question from the SUMI to assess how usable our system was compared to other parking systems.

Method: The participants were presented with the prototype interface system on a laptop computer. The buttons on the screen were selected using a laptop touchpad. Each participant performed 4 different tasks, which is described in the following section. The tasks were presented in a random order and the task completion time and accuracy of each of the tasks was recorded.

After the testing, the participants were asked to answer the questionnaire mentioned above.

Description of the tasks:

1) Pay for a newly parked car
Instruction: You just parked in parking spot A3 and will leave your car parked there for 3 hours. Pay for parking using a credit card.

2) Add more balance to a previously parked car
Instruction: You just got a text message saying that you are about to run out of time in spot K6. You want to add two hours of time. Use Cash.

3) Pay with a parking validation and get a refund
Instruction: After parking your car in spot L4, you ate at a restaurant nearby and received a parking validation (yellow card). Now you’re back at the paying station and you want to get a refund.

4) Get a refund
Instruction: You just arrived back at the lot, and had parked in M2. You got back early, and want to get a refund for the extra time you paid for.

Results: The overall performance of the participants was satisfactory. All of the participants were able to follow the instructions in less than 39 seconds and their average task completion time was 22 seconds (St. Dev.=4.7). The task performance time was usually the longest according to which task they performed first, indicating there was a learning curve. There was a significant individual difference in average performance time between the fastest (14 sec) and the slowest (28.5 sec). The participants were fastest at performing the ‘Get refund’ task which required the least input; entering an identification number.

<table>
<thead>
<tr>
<th>Pay for new spot</th>
<th>Add time to spot</th>
<th>Use validation</th>
<th>Get refund</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>24.57 (St Dev=11.83)</td>
<td>25.84 (St Dev=6.96)</td>
<td>25.06 (St Dev=7.98)</td>
<td>13.98 (St Dev=6.07)</td>
<td>22.36 (St Dev=4.7)</td>
</tr>
</tbody>
</table>

Task Completion Time (in seconds)

While we were satisfied the numbers we have gathered, we paid careful attention to the verbal responses of the participants. A number of participants reported they had trouble with the time selection screen because the numbers were written only in digits (e.g. 3:30) and preferred them to be worded (e.g. 3 Hrs 30Min). We followed their advice and changed the time selection screen in our final prototype. Most participants were not satisfied with the responsiveness of the prototype; they wanted their input to be readable on the screen. Given the technical limitations of the software we used, the only proper feedback that could be given to the users was the transition of the screen. The concept of ‘Refillable Cards’, used as a method to give early-leavers refunds was alien to the participants and a couple expressed their confusion. We decided to name these cards as
‘Refund cards’ because we found it difficult to integrate an option that the users could add balance to the card and the matter did not seem practical. We also created physical copies of the actual Refund card, which could be used if further testing was to be done.

However, our hybrid standard questionnaire results indicated that the participants seemed satisfied with their overall performance. Out of a scale 1 to 7 (1-strongly disagree and 7-strongly agree), the participants rated their satisfaction with the ease of the scenario on average 6.18. They also readily answered that they were satisfied with the time that took to complete the tasks (average=6.55) and the instructions were quite helpful (average=5.73). They were also willing to use our system over other parking systems (average=6.09) and did not find the system itself too difficult to use (average=1.64). All standard deviations were smaller than one.

<table>
<thead>
<tr>
<th>Q1. Overall, I am satisfied with the ease of completing the tasks in this scenario.</th>
<th>Q2. Overall, I am satisfied with the amount of time it took to complete the tasks in this scenario.</th>
<th>3) Overall, I think that the instructions and prompts were helpful when completing the tasks.</th>
<th>Q4. I would prefer to use this system over other parking systems.</th>
<th>Q5. Overall, I found this system to be unnecessarily complex.</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.18</td>
<td>6.55</td>
<td>5.73</td>
<td>6.09</td>
<td>1.64</td>
</tr>
</tbody>
</table>

*After-test Questionnaire Results (St Dev <1)*
Conclusion

After analyzing the system and the prototype we developed, we believe we successfully designed a usable system. When we first looked into this experiment, we decided to use weight sensors, so we tried to keep the cost feasibility in mind. After later review, we found that the cost of WSN was much cheaper, so our system is both usable and affordable. We would like to have done more analyses and more methods to better improve the system and gather more usability data, but without adequate data and larger samples, we were not able to attempt them. We decided to focus on a few of the methods we had the means to perform well. The design choices we made and the analysis we did helped us meet all of our goals for our system to be simple and easy, offer multiple payment methods, incorporate validations, offer refunds, and use an sensor network.
References


Appendix

Sample Receipt

PLEASE KEEP YOUR RECEIPT!

DKH Parking Services
Tel: 713-***-***

PARKING SPOT NO.: C8
IDENTIFICATION NO.: 129-1839
LENGTH OF STAY: 5 Hr., 30 Min
PARKING EXPIRES AT: 2:04PM

Date: 04/16/2011 08:34PM
Card Type: VISA
Acct #: XXXXXXXXXX1234

Total Amount $7.00

Leaving Early?
Enter your Identification Number and We'll give you your money back!

Sample Refund Card

Refund Ticket
Come back to our Parking Lot and you can use this ticket to pay for your parking!
Sample Parking Validation

Parking Validation
Return to our paying station and enter your spot no. to receive a refund.