# Concrete Budget: an Intuitive Retirement Planning Tool

by

Julián A. Caballero

S.B. Computer Science and Engineering
Massachusetts Institute of Technology, 2007

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

May 2008

©2008 Massachusetts Institute of Technology.
All rights reserved.

Signature of Author: _____
Department of Electrical Engineering and Computer Science
May 21, 2008

Certified by: _____
Dan Ariely
Professor of Sloan School of Management
Thesis Supervisor

Accepted by: _____
Arthur C. Smith
Professor of Computer Science
Chairman, Department Committee on Graduate Theses

Concrete Budget: an Intuitive Retirement Planning Tool
by
Julián A. Caballero

Submitted to the
Department of Electrical Engineering and Computer Science

May 21, 2008

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

# ABSTRACT

In times where Roth IRA's are condoned by the government, Social Security is failing, and the US economy is considered by some to be in a recession, there is no doubt that Americans need help with their finances and budgeting for the future. Concrete Budget aims to turn the idea of money, which is difficult to gestate to its fluid and abstract nature, into something tangible and easier to conceptualize. The main abstraction used to solidify a person's budget is by the use of goals (e.g. buying a house, paying for a child's tuition) that can be traded off from one to the other in order to understand the effects of savings a little money on X in order to have enough money for Y. Concrete Budget will strive to be simple and aesthetically pleasing in order to be adopted by uncle Joe, grandma Jane, and even that baby brother who always seems to get himself into tight situations.

Thesis Supervisor: Dan Ariely
Title: Alfred P. Sloan Professor, MIT eRationality Laboratory

## Acknowledgments

First, I would like to thank Prof. Dan Ariely, Rebecca Waber, and the rest of the eRationality group – without you all, I would not have learned and understood so much about behavioral economics and the fascinating ideas that make Concrete Budget. It has been very motivating to talk to people about this project, but without your help, I would not have been able to explain to them all the concepts and ideas.

Next, I would like to thank Chris Resto, Devon Biondi, Liz Arnold, Janna O'Neil, Ronda Devine, Sharona Jacobs, Dori Peleg, and the rest of the UPOP crew. You have been my home away from home since sophomore year and I truly appreciate all your help and constant high spirits as well as the good times we have shared.

Finally, I would like to thank my family and friends for their constant support and motivation. My motto for how MIT students see the institute has always been the following: “MIT is a force to be reckoned with, but we all strive to give it our all.” Without your help and occasional distractions, I could not have given it my all and gotten to where I am today.

# Table of Contents

# List of Illustrations

# 1 Background and Motivation

Concrete Budget has not been the first to attempt to solve the problem of long term financial planning (see section 2.1), but we believe it is the first to attempt to engage the user in order to set up goals, provide visual feedback of progress and necessary changes, and even provide a recommendation for managing an investment portfolio.

## *1.1 Existing Solutions*

The following are current solutions that are in production and open for public use. These tools have been used as motivation as well as background research that aids in the design of the User Interface and in order to define most if not all of the user tasks that people will want to perform when thinking about saving for retirement.

### 1.1.1 BoulevardR [1]

BoulevardR, “the Boulevard to Retirement”, provides a free, five minute evaluation online that will ask the user to select from predefined short-term goals (e.g. buy a home, save for tuition, pay for wedding, buy a car, eliminate debt) and long-term goals (e.g. Travel, going back to school, starting a new career, spending time with family.) It will also ask for the user's birth year, gender, marital status, and others in order to estimate how much money to save for retirement (given expected life span, how many people one must support, etc.)

BoulevardR does extremely well at providing an easy to use tool that takes a short amount of time and generates an intuitive interface (in the form of a survey.) Almost anyone who will want to use this product will know how to fill out a web form and will be familiar with drop-downs, radio buttons, and drag-and-drop (this last one is questionable.) One of the first flaws that quickly stuck out was the inconsistency between drag-and-drop of long term goals in order to prioritize them and clicking on short-term goals in order to add them to the set of important goals. Jakob Nielsen's Usability Heuristics [2] state that “consistency and standards” should be kept throughout the program in order to prevent user confusion.

### 1.1.2 MoneyWell [3]

Much like the traditional Quicken and Microsoft Money, MoneyWell provides a personal financial-accounting tool that tracks short-term, recurring expenses. Its main advantage over its competitors in its field is its intuitive and well-developed metaphor of using buckets to represent expenses that need to be paid off. Their use of this metaphor makes it a very interesting research base since it takes away some of the negative feelings people have to accounting and financial planning. A bucket starts out tipped over and opaque meaning it is an expense that has no money flowing into it. It then becomes an up-right bucket in full color, meaning it has some income flowing into it. If an expense is recorded, but there is not enough money to cover it, a red exclamation mark is placed over the bucket and an “overspent” message is presented.

MoneyWell is a Mac-only software, which limits its target audience, but because of the high

standard of code and user interfaces of Mac software, it is by far the most pleasing software tool to use for financial planning. The use of the bucket metaphor gets users thinking in terms they are more familiar with and not scared of. The user interface, which can be seen in Illustration 1, is very sleek and keeps users engaged, without the agony of unpleasant designs that do not consider aesthetics important (which sometimes translates to not caring about the user.)


*Illustration 1: MoneyWell User Interface [2]*

*Note the bucket metaphor on the left and the overall aesthetics of the interface which is consistent with other Mac software such as iTunes and Finder.*

### 1.1.3 myPlan Snapshot by Fidelity [4]

The fight for retirement planning is not just for the little guys: Fidelity's myPlan Snapshot is very much like BoulevardR in terms of the ease of use and low time commitment to see a result. The main difference is that, as anyone would expect from any major investment banking firm, the tool is mainly designed to suggest how much one would need to save to retire, and then focuses on how to balance savings between stocks and bonds (see Illustration 2.)

An interesting feature of myPlan, which is unlikely to be in any desktop software, is a talking man who guides the user through the survey, giving it a more personal touch as well as making it enjoyable to do. It is unclear whether or not the voice is interactive or if it is queued as survey pages are rendered, but it almost seems as if you are talking to a Fidelity representative who is trying to determine the best plan for you.

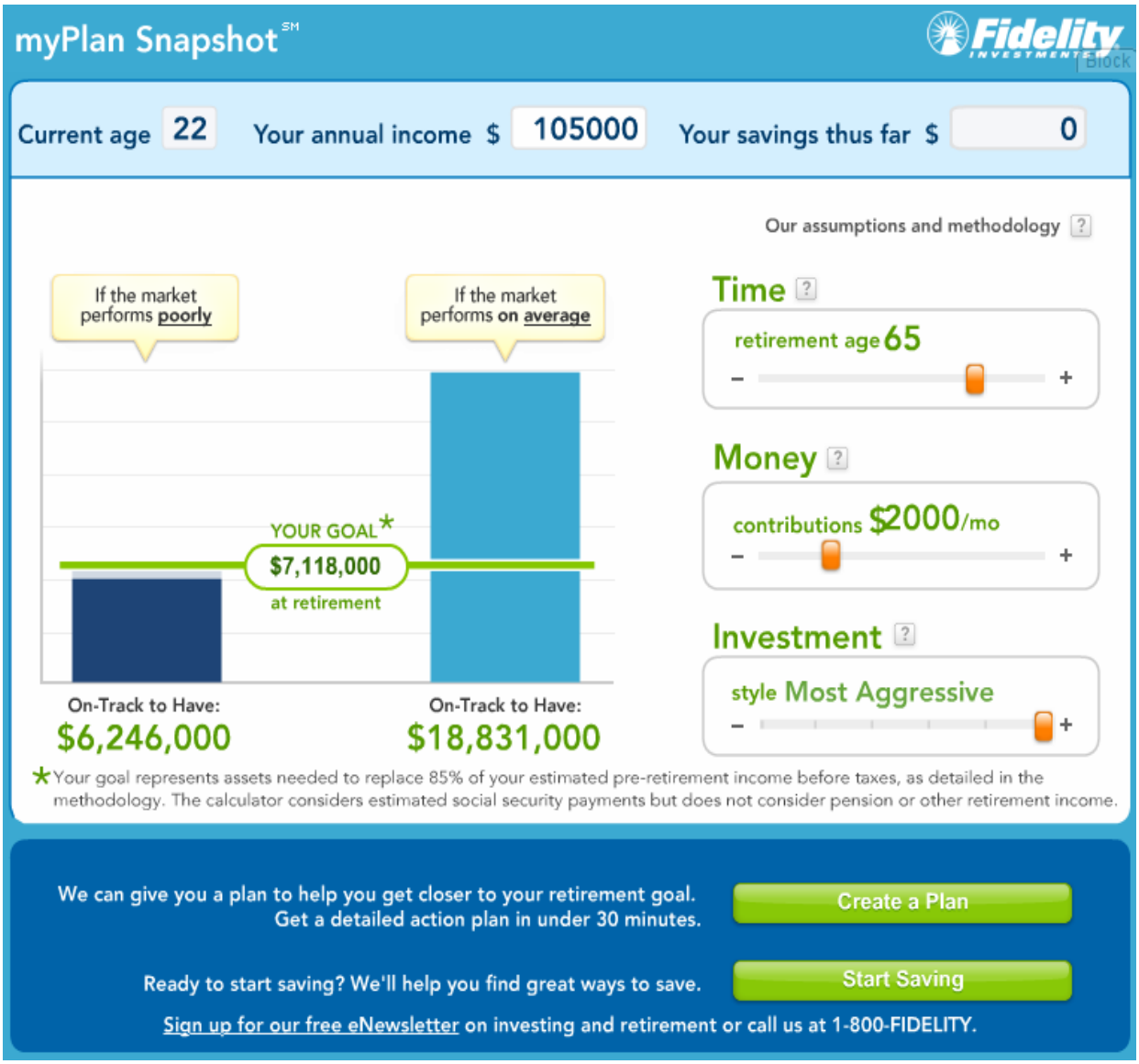


*Illustration 2: Fidelity's myPlan Snapshot [4]*

*This image is a snapshot of the resulting view that appears after answering several demographic questions as well as the current saving strategy. All but the retirement age were questions that were answered previously.*

### *1.2 Motivation*

The greatest motivator for Concrete Budget is people and their irrational thinking of financial planning. There is a high demand for a simple and friendly tool that will aid everyone in order to reach their life goals. The current financial solution to planning for ones future usually involves a spreadsheet application and too much complicated mathematics for the everyday person.

User Interfaces are one of the most important components in software, although it is not until recently that marketers and companies have understood this; no matter how good an algorithm is, if it takes the user too long to understand how to use it or if its usage causes discomfort, the software is doomed to stay with the users who care more about the outcome than the extenuating process they must go through to get there.

# 2 Initial Plan

Good user interfaces are developed through an iterative process, taking user input throughout most of the steps. This guarantees the users will like the tool and will find it useful. Fortunately for us, everyone is a user – and even better is the fact that most adults over twenty do not need much persuasion to test or give feedback when it comes to software that will help with future financial planning.

## 2.1 User Interface Design and Implementation

The design and implementation of the user interface was intended to be broken up into three stages that were to be run iteratively in order to have low fidelity, low depth, and high breadth model in the beginning, and culminating with a high fidelity, high depth, and high breadth tool that can be presented as a proof of concept [5].

First, one must start with a low fidelity, low depth, and high breadth model in order to cover all the potential tasks a user will want to perform, but keeping the design minimalistic so as to make changes easy and cheap (cost is mostly in terms of time, not money.) This is accomplished with paper mock-ups of the software, showing how views lead to other views and how to divide or join views according to the common mental model. These can be free-hand drawings which give a general look and feel for the user interface, but are not yet burdened by details such as colors and sizes. The main goal of this prototype is to ensure all user tasks that have surfaced are accounted for and that no big-picture problems should arise after this round of testing.

Next comes a high fidelity, low depth, and high breadth model which brings to focus the major details of the user interface such as hue selection, alignment of objects, and font of text. This model is accomplished with computer mock-ups which do not perform any of the actions, and are merely to present the look and feel of the user interface in a much more detailed manner. This step allows us to separate performance problems and background implementations from what is truly important to the user (mainly, what he or she can see.) Since most of the breadth issues (e.g. how many windows are needed, what can a user do) have been resolved in the previous step, this

allows the designer to get user input as to how to make the design look better and be more intuitive.

Finally, a high fidelity, low depth, and high breadth model was generated in order to find all the minor details that a user is uncomfortable with or would like to see done differently. Usually, because most of the aesthetic problems were solved in the previous step, this milestone allows for testing the interaction between views and how the user feels when using the software as a whole package. This prototype is accomplished with actual code that can run autonomously and should be as close as possible to the final product as possible, only requiring minor tweaks that need to be made in order to solve the issues that arise in this final prototype.

# 3 Work Plan

This project requires the design, development, and testing the User Interface of a Java desktop-platform system that will aid and educate consumers in retirement planning and any trade-offs they will have to make. The design required iterating through prototypes, from pencil and paper, all the way to a fully implemented Java interface. This will then be tested by potential users and will be re-worked to improve on the design with the user's input, following the user-centered designed model.

The work comprises the activities and corresponding estimated effort as depicted in Illustration

3. Note the breakdown is a logical rather than sequential breakdown.

Due to the time restrictions of all involved, testing will be left as future work in order to finalize the tool and make it as intuitive as possible to the users. Some minimal testing was done as a sanity check, but a full user-test should be conducted in order to verify the results from this small subset.

| Phase | Activity Number | Description | Estimated Effort (hr/ week) | Deliverable | Estimated Date of Delivery |
|---|---|---|---|---|---|
| 1 | 1 | Initial design of the overall program structure and individual pages.<br>a) Design of overall work-flow of a typical user<br>b) Design of individual screens with which the user will interact | 5 | Paper prototypes of both designs | October 1, 2007 |
| | 2 | Meet with Rebecca Waber and Dan Ariely to refine design | 2 | | |
| 2 | 3 | Secondary mock-ups of the individual pages with more detailed views of screens | 10 | Images of screens with colors and more true look-and-feel | October 22, 2007 |
| | 4 | Meet with Rebecca Waber and Dan Ariely to refine design | 2 | | |
| 3 | 5 | Coding of first working implementation of designs | 10 | First revision of | 4 weeks from |

| | | | | | |
|---|---|---|---|---|---|
| | | | | Java front-end | second delivery |
| | 6 | Meet with Rebecca Waber and Dan Ariely to refine design | 2 | | |

*Illustration 3: Work Plan for Principal Investigator starting October 1, 2007*

*Work accomplished between September 5, 2007 and October 1, 2007 was solely for introductions and passing of knowledge – no clear deliverable can be shown from these meetings.*

# 4 Design and Implementation of User Interface

The design for Concrete Budget started as drawings on pieces of paper and culminated in a Java Application with high breadth, depth, and fidelity. The following section shows the different phases of the design process for our tool.

## *4.1 Design of Paper Mock-Ups*

When initially designing something, one must start with cheap prototypes that can easily be thrown out or greatly modified. For this reason, paper prototypes were chosen as the first iteration in the design process in order to come up with as many ideas as possible in a short

amount of time. The idea behind a paper prototype is to try to come up with as many possible solutions to the problem as possible, without spending too much time on implementation and allowing cheap and simple modification to be made on the fly without expensive ramifications (as with the Waterfall Model [6] in Software Development.)

Paper prototypes are designed to gather data on the following [7]:

- Concepts and terminology – test if users understand descriptive or key words
- Navigation / work flow – the navigation for user interfaces should allow users to perform the tasks they want to do in a manner that is effective and efficient
- Content – possibly the most important of all points, content must be tested in order to determine if anything is missing or too detailed; if a user cannot find what he or she is looking for, something needs to be changed
- Page layout – although it may be possible to perform every single task the user wants to, if the user cannot find how to do something because the layout is not intuitive, page layout should be studied and possibly changed
- Functionality – sometimes, user tasks come up during the development process that were not
- defined previously; if the user wants to perform some task that is not possible with the current interface, the lack of functionality could prove to be the sinkhole that makes a project fail

The design for Concrete Budget began with the transfer of knowledge from Dr. Ariely to the

principal investigator, initially driven with the idea of money as water (for its fluidity) that would fill different goals (to be thought of as buckets.) This analogy would have income or savings seen as the source of water, which would be distributed among all the buckets in hierarchical order according to the user's priorities. This would require the user to think which goals are actually necessary, which goals he or she is interested in, and which goals are mere wishes that would be nice to have. Another key idea in Dr. Ariely's model is the use of trade-offs between two or more goals: think of it as trading off between buying a smaller house in order to pay for a child's tuition at a more expensive college. Illustration 4 shows the initial idea behind trading off between two goals, using a slider to manipulate the goal amounts.

### 4.1.1 Two-Goal Trade-Off

As can be seen in most paper mock-ups, Illustration 4 shows the original idea of buckets of water that will change size depending on the target amount of a goal (e.g. a house costs $700,000.) This type of prototype allows for quick and easy annotations, as can be seen in Illustration 4. Because it is a pencil and paper design, the designer can easily make changes if an idea was misinterpreted or needs fine-tuning.

The idea behind the design seen in Illustration 4 was to allow users to select which two goals to trade off and allow them to fix certain parameters such as time and target amount. This prototype allowed us to see the complications that arise when comparing two goals as well as the technology necessary to make a simple and intuitive user interface. The concept of trading off

goals is relatively new and is extremely powerful because it allows users to see how saving a little money on one goal, can go a long way towards reaching the target of another goal.

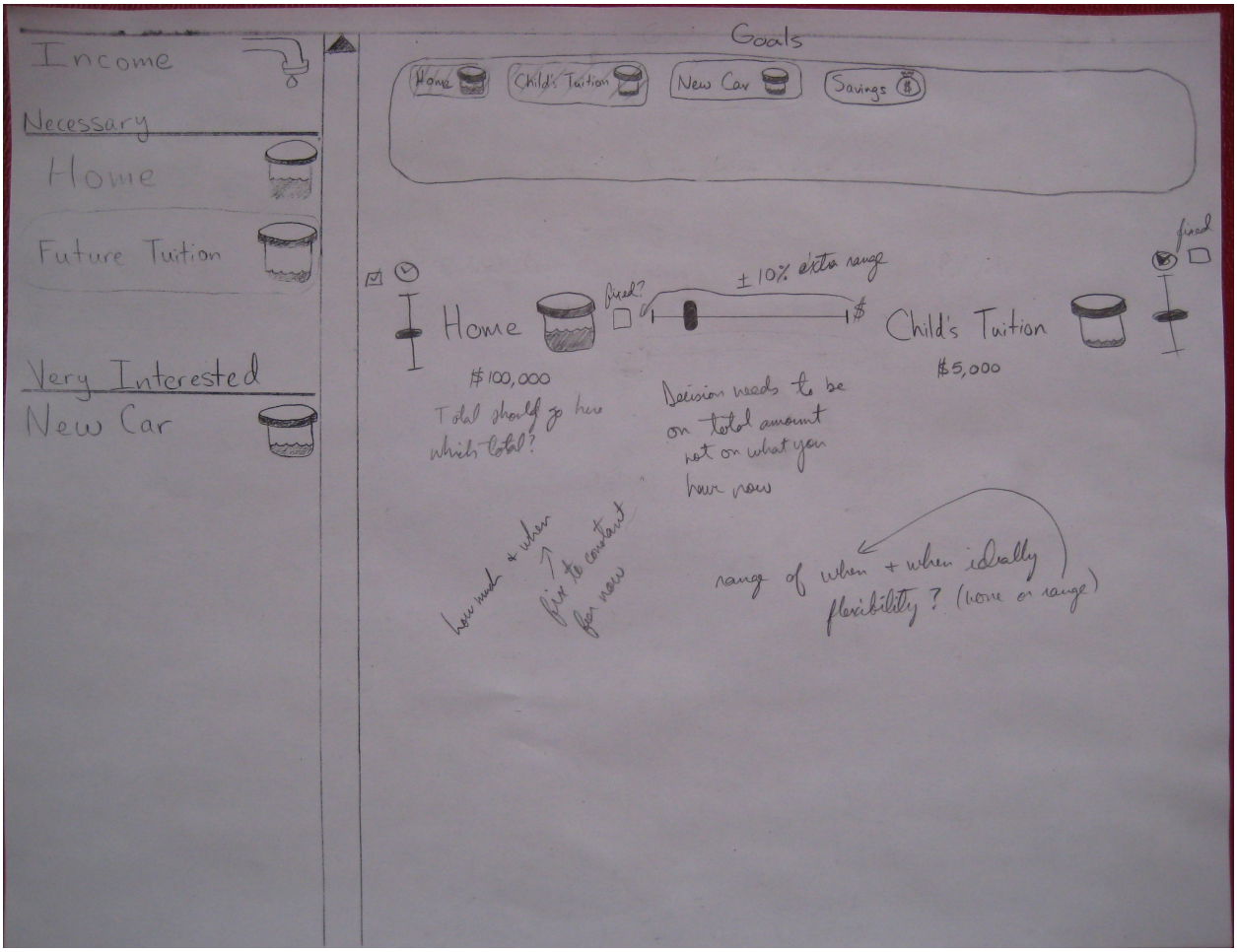


*Illustration 4: Paper Mock-Up, Two Goal Trade-off*

*This Illustration shows the initial design idea behind the interface that would allow trading off between two goals. The slider in the middle would accomplish the change in amounts. There are many annotations, as this is a paper mock-up.*

### 4.1.2 Multiple Goals Trade-Off

Another important screen designed using a paper prototype was the one that allowed multiple goal trade-offs. Illustration 5 shows many different ways to represent information as well as ways to manipulate data: it was originally unclear how users would change the target amount of each goal in the Multiple Goals Trade-off screen, so the mock-up in this Illustration was designed in order to show to different users and see which they would prefer. Some are simpler than others; some show detailed, numeric information.

As an example, we will study the middle design in Illustration 5: a slider would allow users to change the target goal amount and two icons on the right of the bucket would serve as flipper buttons that would increase or decrease the target goal amount. The icon that would increment the target goal amount would have a small bucket transforming into a larger bucket and vice versa for decrementing. Unfortunately, this design would take up too much space, would be confusing for the user (we also use changing of bucket size to show the size of the goal – there are three ranges of amounts which produce three different sized buckets for goals.) This type of prototyping proves to be a great solution to preliminary design problems such as how to present data to the user.

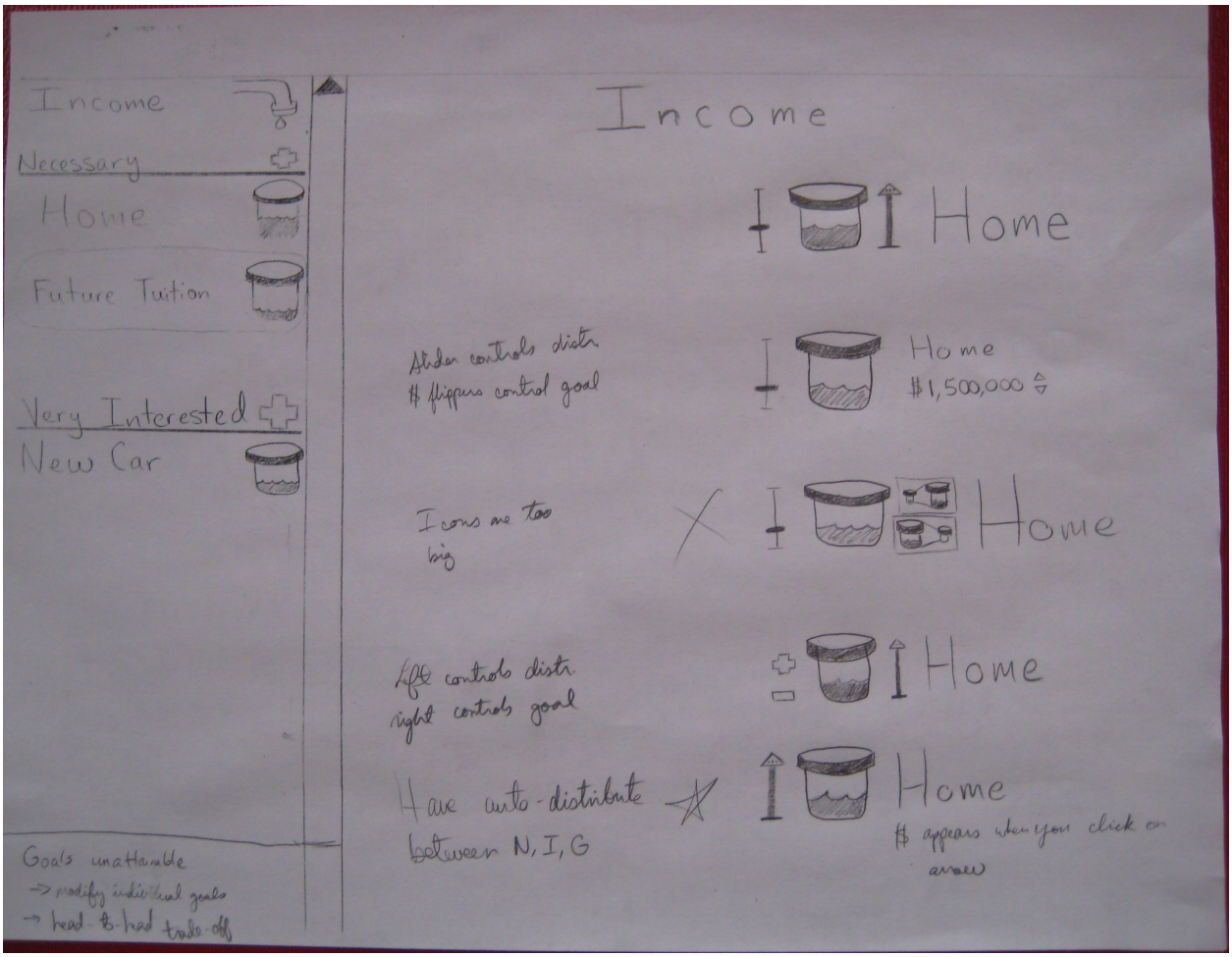


*Illustration 5: Paper Mock-Up, Representing and Manipulating Goal Data*

*This design was an attempt to determine which type of slider or flipper would be the simplest and most intuitive to the user so they could increase or decrease the amounts of each goal.*

### 4.1.3 Income Pool

Illustration 6 is a good example of designs that are discarded after implementation, but because of the low cost of creating these designs, it is almost a gain since very little time was spent on it in order to decide not to spend a significant amount of time implementing it. In this case, the

user's income was to be thought of as a pool of water with a spigot which would be the source of water for all the buckets. Although it is a good analogy for where users will get money for their goals, it is somewhat unnecessarily complicated since users can simply distribute their savings. Also, the pool analogy, although logical and fun for the user, is not necessarily important and does not add any value since the user knows where the money is come from.

### 4.1.4 Goal Details

The last of the paper mock-ups is probably the one with the most detail, but the one that was most instrumental in the design for Concrete Budget. Illustration 7 shows the detailed goal view of paying for a child's tuition. The simplistic design behind this mock-up kept in mind the important information users would want to know.

This prototype was very helpful, especially to determine the best terms to use. Lowest Acceptable Total, as seen in the illustration, was later changed to Minimum Necessary, which is much more descriptive and understandable. Also, even though the back-end will take into account user risk and will try to help the user depending on their risk level, since risk is such an intangible factor, the prototype allowed us to realize that simply asking for “Minimum Necessary” would be enough to gather the important information regarding the user's risk without having to ask the user explicitly for this information (especially since he or she cannot easily materialize it.)

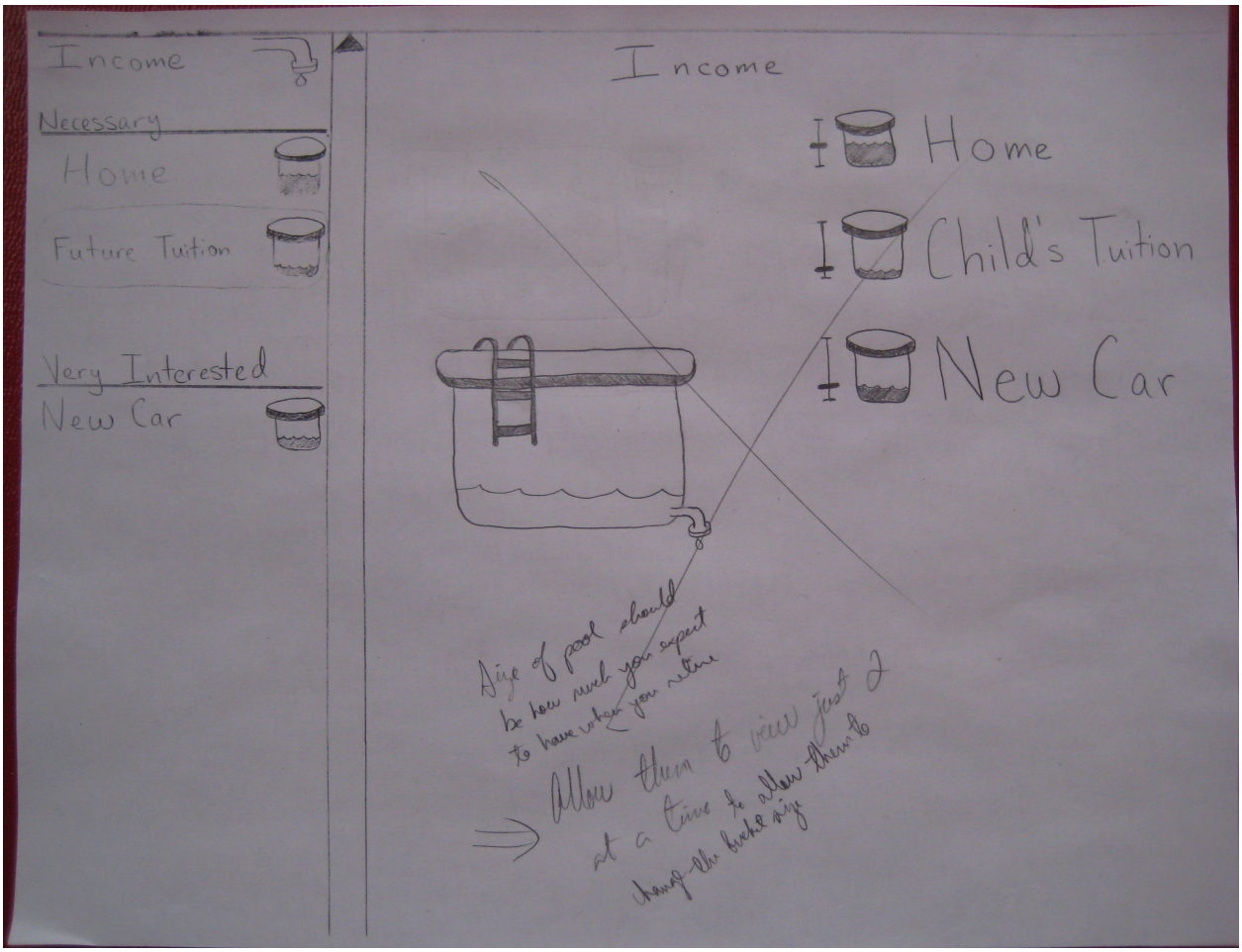


*Illustration 6: Paper Mock-Up, Income Pool Distribution*

*The idea was to have users think of their income as a pool of water that would get distributed to all their goals. This design was discarded due to the complications that emerged, such as accounting for day-to-day expenses and other expenses that do not go directly towards goals.*

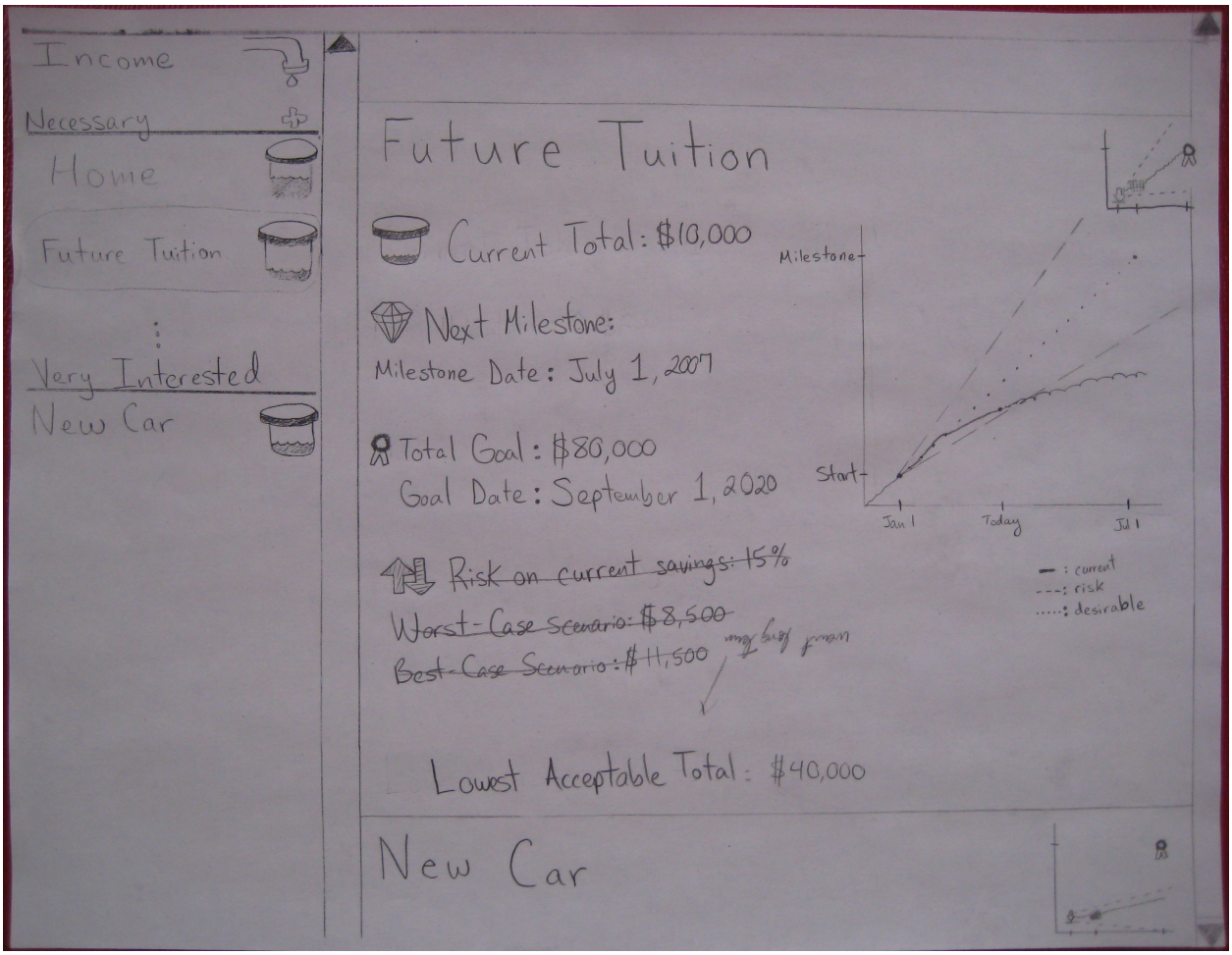


*Illustration 7: Paper Mock-Up, Goal Details*

*This design is probably one of the most important screens as users will be looking at their specific goals in order to see how they are doing and how much they need to save in order to reach all their milestones and deadlines.*

## *4.2 Design of Wireframe Prototypes*

Once paper prototypes are analyzed and modified, a more accurate design is necessary in order to incorporate more of the medium to small grain details such as component interaction, general color and typography, and revalidation of the data gathered from the paper prototypes. This mock-up is generally considered to be of medium to high fidelity, high breadth and low depth.

All wireframe prototypes were made with the GNU Image Manipulation Program (GIMP) http://www.gimp.org/, an open source image editing tool similar to Adobe's Photoshop http://www.adobe.com/products/photoshop/family/index_ps.html.

### 4.2.1 Two-Goal Trade-off

The paper prototype for this design allowed us to discard superfluous information and emphasize the important information. Allowing the user to fluctuate the target date of goals was eliminated as a trade-off for simplicity and clarity for the two-goal trade-off screen. As can be seen in Illustration 8, the wireframe prototype is much more simple, focusing on the main task users want to perform in this screen.

From this point on, Dr. Ariely thought changing from a water and buckets analogy to a fuel and

gauges analogy would prove to be more helpful to users, so you will see the change in icons to adopt this new model.

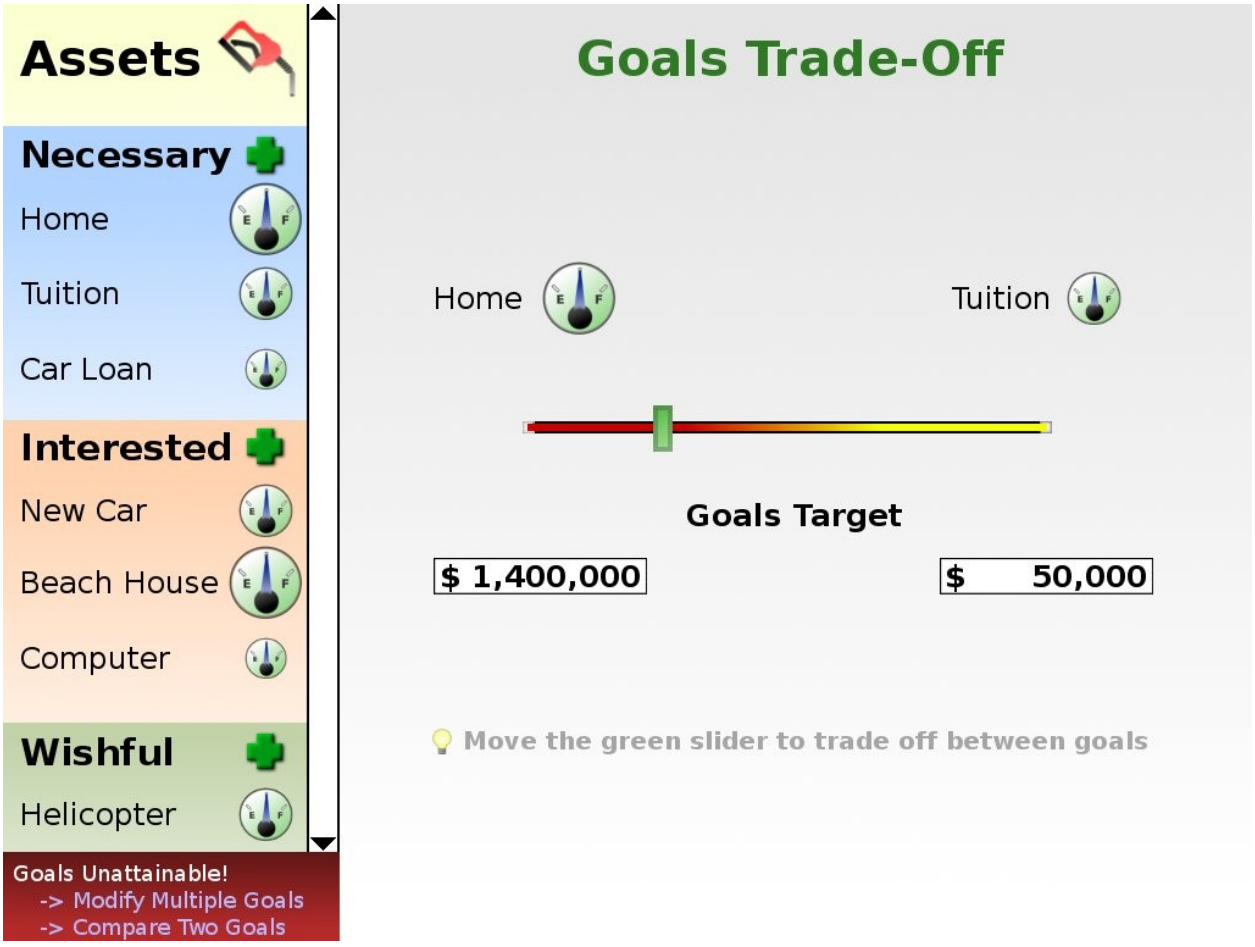


*Illustration 8: Wireframe Prototype, Two-Goal Trade-Off*

*In comparison to the paper prototype, this design is much more polished and has many more details that start to come into play during this stage of the process.*

From comparing Illustration 8 with its counterpart, Illustration 4, it is easy to see that a

wireframe model is much more true to the final design, thus the high fidelity, but because it is simply a static image, it maintains the same level of depth as the paper prototype. In designing this wireframe, we took into account the color schemes, such as the ones for the fuel gauges, which play an important part of the tool. Although Illustration 8 only shows green fuel gauges of varying sizes, these are meant to be green or red depending on whether the user is on track or not for that particular goal. There was some debate with what constituted as being “on track”, but in the end we decided to make it be a matter of having enough savings to meet the current milestone for that goal. The fuel gauge needle, which shows how much “fuel is in the tank”, is used to represent how much money has been allocated to this goal. The needle starts pointing at “E” for empty, and once the entire amount for the goal is saved, it will point to “F” for full.

### 4.2.2 Multiple Goals Trade-Off

Once the major kinks are worked out, wireframes try to give a very similar look to that of the final product; Illustration 9 shows an almost final draft of the Multiple Goals Trade-Off screen.

This design was chosen because of its numerical nature: users want to know more or less how much they have to spend on something, so the flipper is the best in order to give users the ability to visually see how their changes affect the distribution of their savings. For example, being able to see that reducing the amount to spend on a house can allow a user to be able to pay for three other goals, this will most likely encourage them to do so until some extent.


*Illustration 9: Wireframe Prototype, Multiple Goals Trade-Off*

*The wireframe shows the different goals in their respective priority categories. In this screen, users see how changing one goal affects the likelihood of being able to attain a different goal.*

This design shows a more "realistic" set of goals, in that users will most likely encounter when they input all their goals, they will not have enough money to achieve them all. In this case, the last two goals are not on track to being accomplished, as can be seen from the red gauges.

### 4.2.3 Assets and Debts

Instead of having a pool of income, we now have a simple “Assets and Debts” screen in which users enter their financial information. This can be changed at any time by the user in the event of changes, preferably when the user saves more money for future goals. Illustration 10 shows the simple design that is meant to be used by the user during his or her first time using the tool and whenever he or she needs to make changes to that information.

All the icons, created by the principal investigator, were designed specifically for this tool in order to make all fields easily browsable. In today's world, where speed and efficiency are key, icons and indicators help users quickly identify what they are looking for. The icon for the “Loans” field was not created until the final implementation, so it is not on Illustration 10. The colors follow the same scheme as in the rest of the mock-ups: green for something positive (e.g. on track to attain a goal,) and red for something negative (e.g. not having enough money to pay for a goal.)

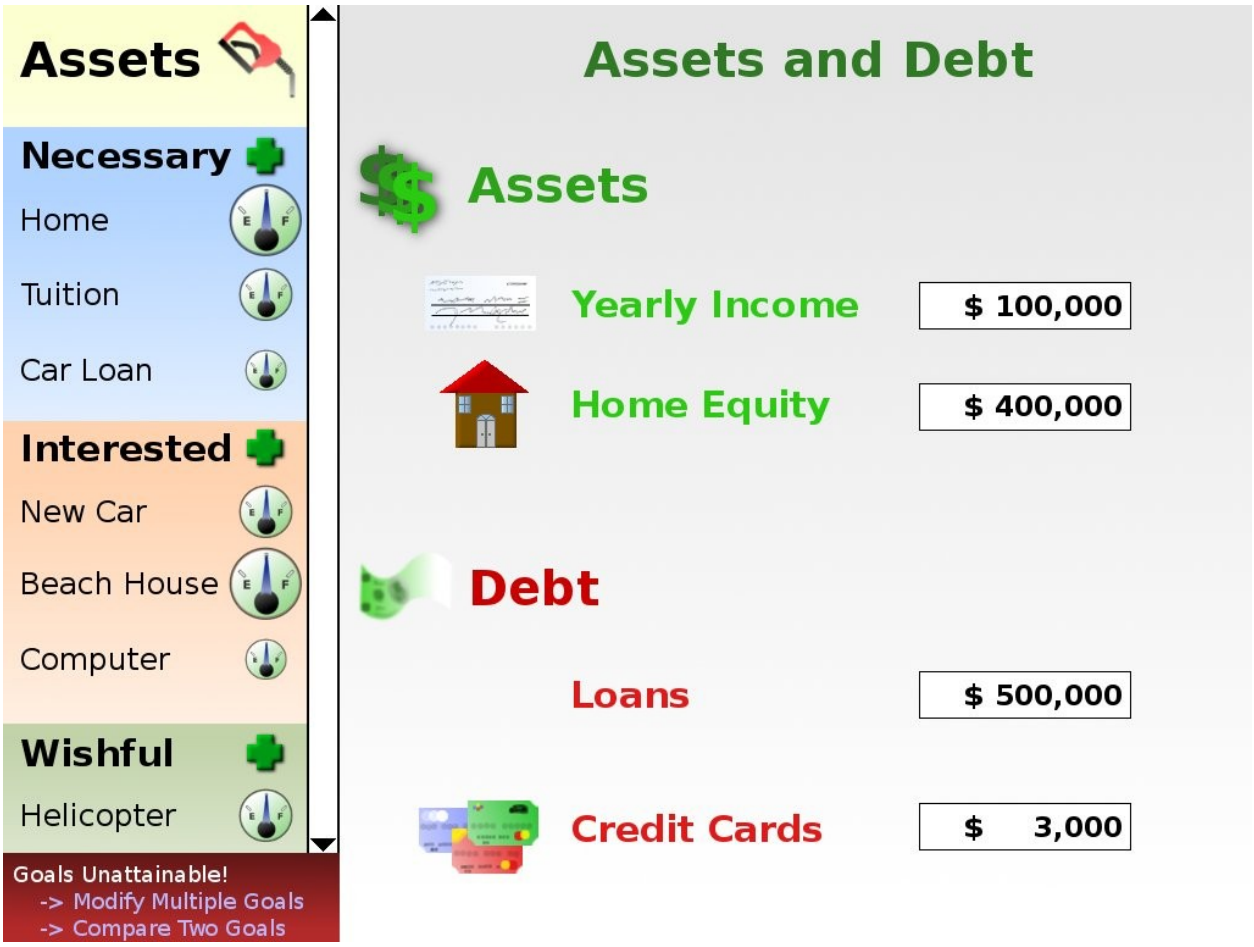


*Illustration 10: Wireframe Prototype, Assets and Debt*

*This prototype attempts to be simple and descriptive to the user so that he or she can easily and efficiently complete the desired task. The icons and font color are meant to help find the fields.*

### 4.2.4 Goal Details

The goal details design did not change much from paper prototype to wireframe mock-up, but now more consideration was put into the details, such as colors and icons. Illustration 11 shows

the wireframe model used to validate the simplicity and effectiveness of the goal details design.

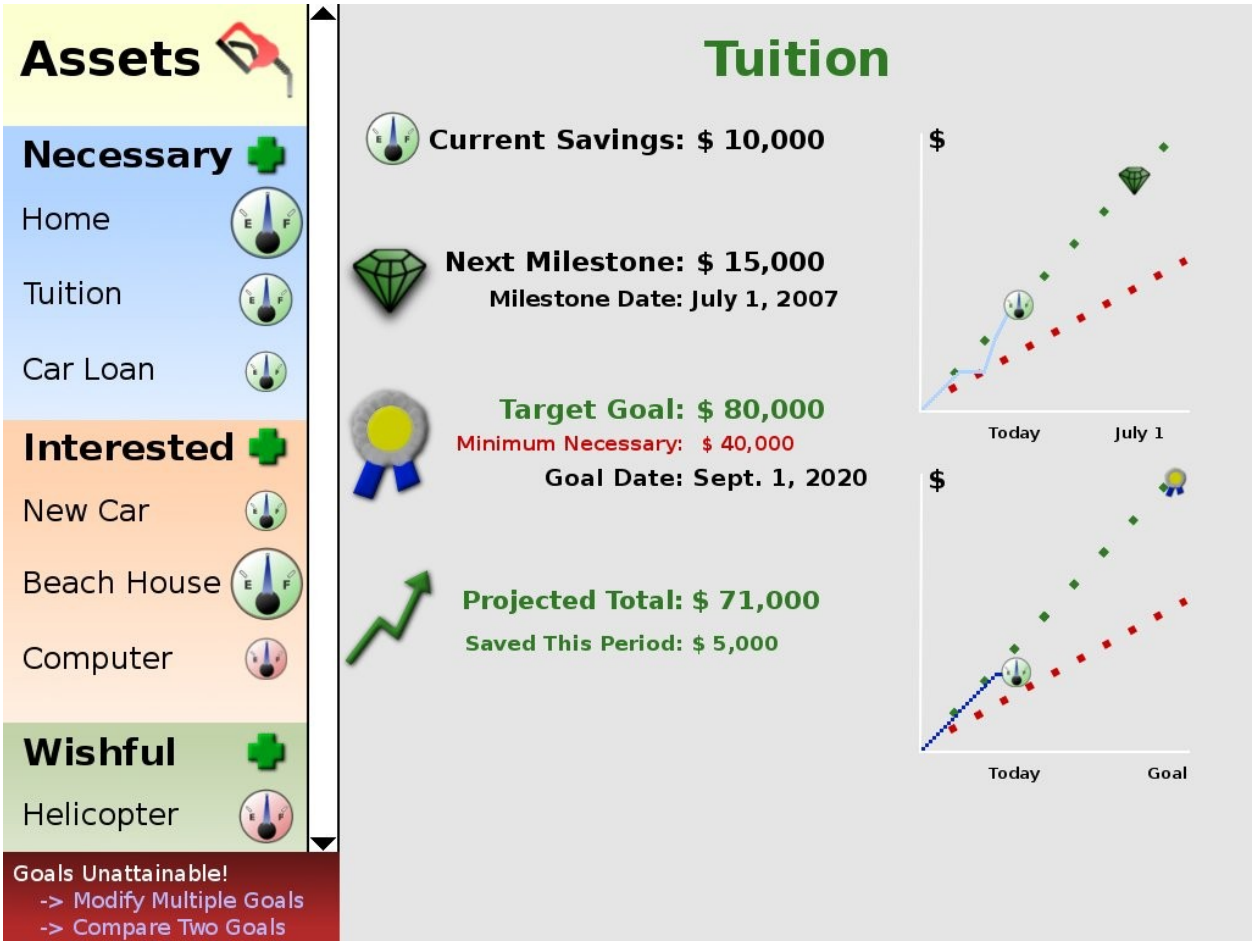


*Illustration 11: Wireframe Prototype, Goal Details*

*This design was modeled after a hypothetical goal of paying for a child's tuition in over ten years. The milestone amount and date are not representative of the actual milestones algorithm.*

The goal details screen will probably be the most viewed page by the user since he or she will want to check up on specific goals and see what can be done in order to make goals more

attractive without making them unattainable It represents numerical data in a graphical way as well as with text, since the graphical method does not provide clear numerical values.

In this screen, we have kept with the color scheme of green being a positive indication and red being a negative one. In this particular example, the target amount would always be green, since it is what the user is attempting to achieve, the minimum necessary would always be a red amount, since it indicates the threshold, and the projected amount would be either green or red if the user can or cannot achieve his or her goal at the current saving rate. The graphs on the right represent the milestone (short-term) data and the overall (long-term) data. The green dotted lines in both cases represent the steps that would lead to reaching the target, whereas the red dotted lines represent the steps that would lead to having the minimum necessary.

The icons in this screen are also used to help the user identify the section he or she is interested in without having to read all the text. Because the icons are mostly different, it would be easy to find the section the user is looking for once it memorizes what the different icons represent – this memorization step will only have to be done once, since the user should be able to remember what the icons stand for and go almost directly to them when looking for certain data. The green gem is used to represent a milestone and the gold medal is used to represent the target goal – these icons are shared in both, the detailed text, and the two big-picture graphs.

### 4.2.5 Personal Information

Although Concrete Budget is meant to be a research tool to understand human rationale when it comes to financial planning, we hope it will ultimately end up as a financial tool provided by big players such as Bank of America, Fidelity, or others in order to help as many people as possible. For this reason, a Personal Information Design was created, in order to think about the whole picture, although not necessarily including it in the final tool for this project. Illustration 12 shows this design, which follows the same general scheme in Concrete Budget.

This design is by no means a final draft: it is meant as a guide to whomever will work on this in the future that wishes to use information like this for his or her tool. The main reason this information is important is to be able to make an educated guess of life expectancy, family contribution and expenses, and risk. The zip code is helps narrow down the population to a certain area in the US (although this could be changed to be internationalized) so as to get a better sense of the person's lifestyle.


*Illustration 12: Wireframe Prototype, Personal Information*

*This prototype is meant to think about future possibilities of Concrete Budget, where family information will be very useful in providing suggestions and recommendations.*

### 4.2.6 Goal Creation

All was well and good before, but we had neglected the fact that people will need to CREATE goals as well as manage them. This is one of those screens that is very important, but not very

commonly used, since users will only need to create a goal once and will most likely create all their goals in the first few times using the tool, and then will wait possibly years until they have to create a new goal. Illustration 13 shows the goal creation wireframe design.

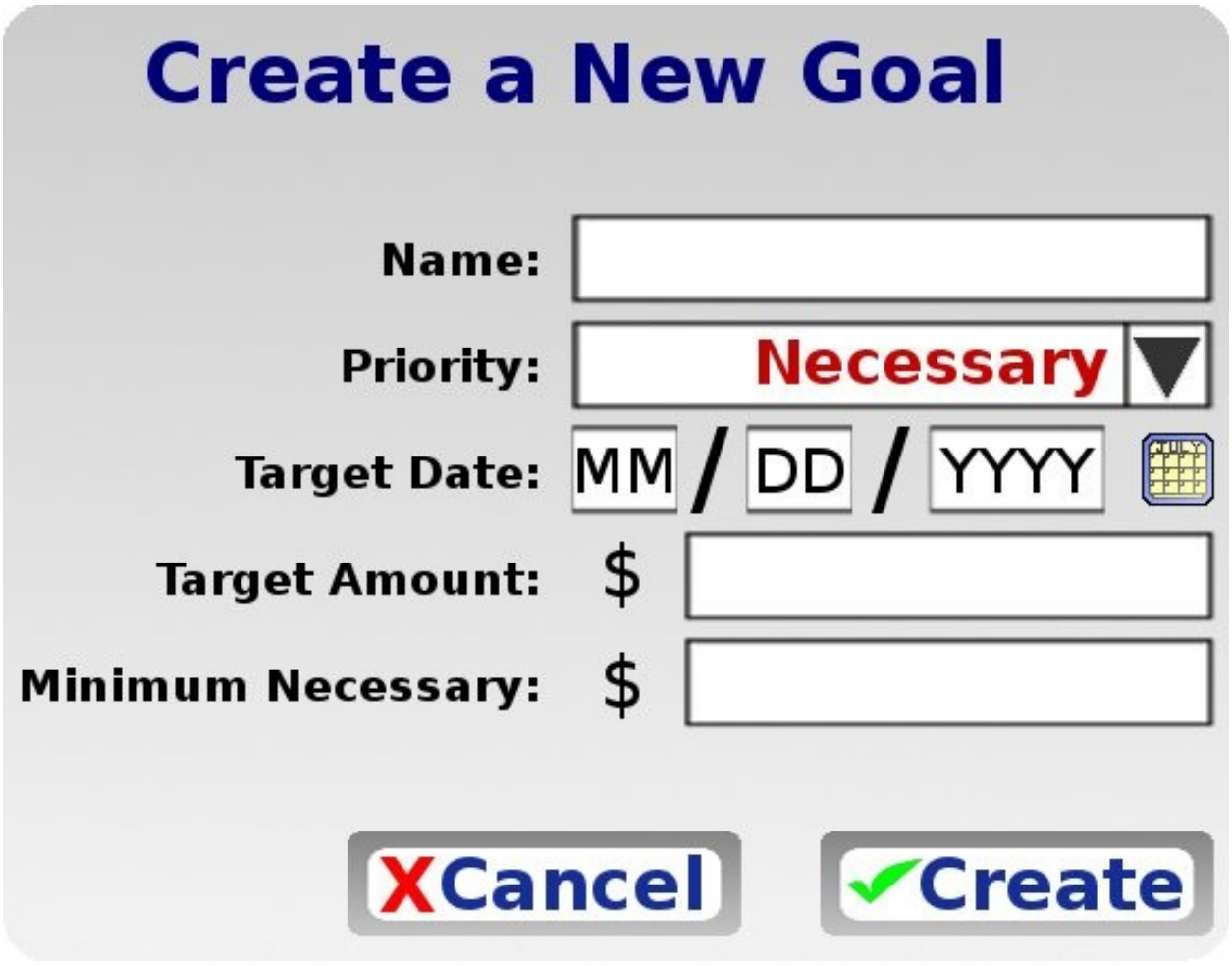


*Illustration 13: Wireframe Prototype, Goal Creation*

*This screen was meat to be a pop-up that would appear when users click on one of the green plus signs next to each priority label. It is meant to be quick and simple for users to make a new goal.*

The idea behind this screen was to make it a pop-up that would appear or slide out whenever a user clicked on one of the green plus signs next to the respective priority label. The user would then enter the appropriate information, change the priority if necessary (if, for example, a user clicked on the green plus sign next to the "Necessary" label, it would automatically use that as the priority), and then create the new goal.

Although the user may not be aware of it, this innovative way to define a goal can actually help with determining the user's risk level. By asking for the minimum necessary amount for a goal and the date of the goal (either of which could change,) one could approximate the user's risk with regards to this particular goal. For example, if the minimum necessary is very close to the target amount, the user is more risk-averse about this goal than say, if the minimum necessary was actually half of the target amount. As a more concrete example, imagine the different risks between paying for a child's tuition as opposed to the risk of buying a new car. In reality, most people do not need to buy the most expensive car on the market, but if they are not aware of how much it could help them reach their other goals, they will probably try to buy the most expensive car they think they can afford with their current expenses.

## *4.3 Design of Software Interfaces*

The final software prototype was designed and coded in Java using a combination of Eclipse (http://www.eclipse.org/), NetBeans (http://www.netbeans.org/), and JFormDesigner (http://www.jformdesigner.com/), in order of most used to least used. This section will only

present the user interface prototype for the software. For more information on the actual code and back end behind Concrete Budget, see section 4.4.

Software prototypes are the most complete of all, with high fidelity, breadth, and depth. It is meant to be the final draft of a design, but must leave room for change in case any significant changes need to be made. This is a somewhat complete working model, missing several of the components in the wireframe model in order to improve simplicity, functionality, or because there simply wasn't enough time to implement it. Also, relying an on an undergraduate to help with the code was somewhat problematic due to the lack of team-work and collaboration experience.

### 4.3.1 Two-Goal Trade-Off

Since most of the important additions from a wireframe design to a software interface are in the back end, Illustration 14 shows that the software interface for the two-goal trade-off is very similar to that of the wireframe. During the creation of this interface, we realized a need to undo changes by the user, thus adding a “Reset Amounts” button which would set the amounts back to what they were before the user entered the trade-off screen.

The user can move the slider from one side to the other, which would trade off between the two goals, keeping the total amount of the two goals the same, but passing it from one goal to the

other. As the goal amounts change, so do the fuel gauge icons; there are three tiers of goals: small, medium, and large. Small goals are considered to be below $5,000, medium goals are considered to be between $5,000 and $100, 000, and large goals are anything bigger than $100,000. As the user moves the slider, if any of these threshold were reached, the icon would change size. Also, because reducing a target amount can decrease the amount needed for the current milestone, if a change made the current milestone reachable, the icon would also change colors from red to green.

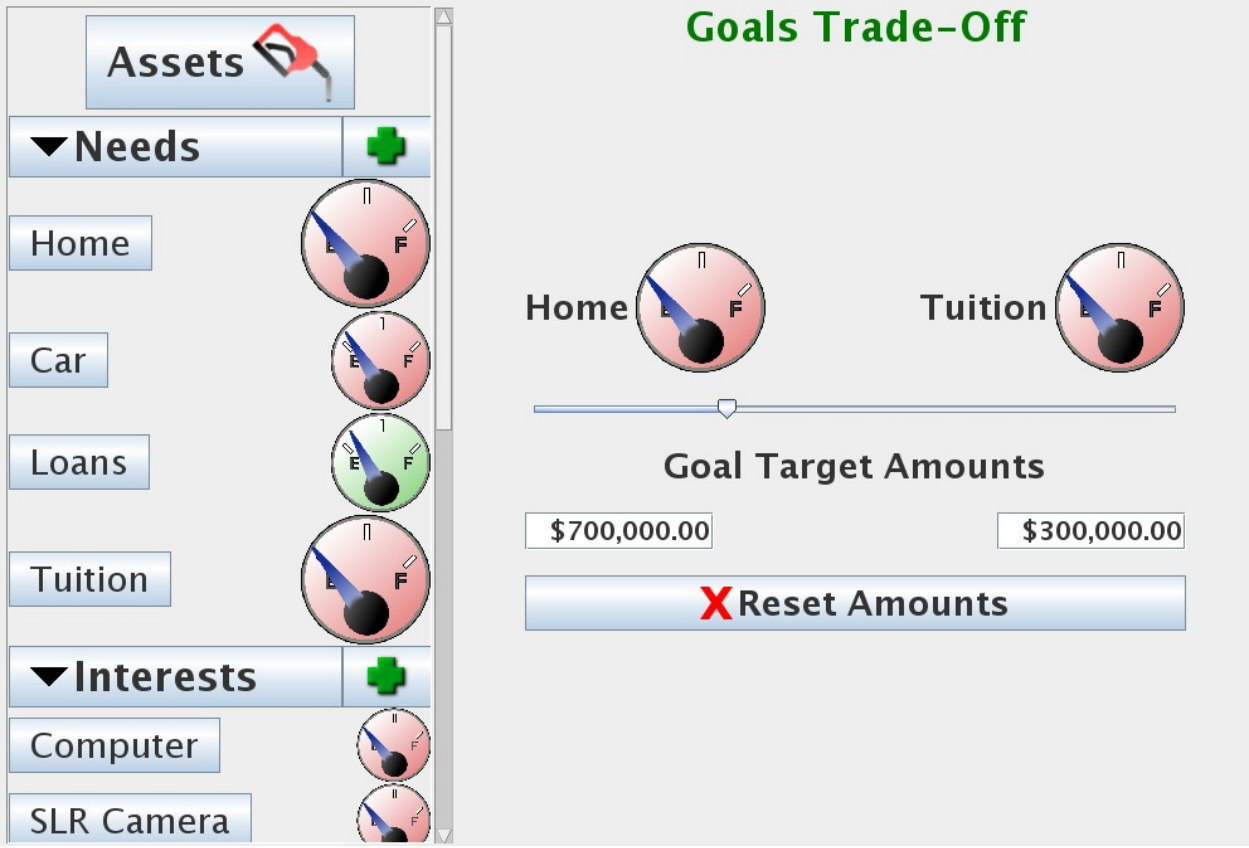


*Illustration 14: Software Interface, Two-Goal Trade-Off*

*Although very similar to the wireframe counterpart, we were able to identify the need for an undo button, in case the user wants to go back to the original values.*

### 4.3.2 Multiple Goals Trade-Off

This interface is fairly simple and is meant to allow the user to view and modify all goals in one screen. This is useful for improving the speed with which users can perform their desired tasks, although it is more likely that users will want to trade off between two goals since this makes

more logical sense and is less complicated. Illustration 15 shows the sample profile in the multiple goals trade-off panel. The different priority labels are colored in order to show urgency (thus, “Needs” is red, “Interests” is green, and “Wishes”, not visible, is blue.)

As with the Two-Goal Trade-Off interface, the multiple goal version updates the goal icons “on the fly” as the user modifies the target amounts. If a goal passes the size thresholds defined in 4.3.1, the icons will change size; if the current milestone is currently reachable with the new values, the color of the icon will change as well.


*Illustration 15: Software Interface, Multiple Goals Trade-Off*

*Similarly to the Two-Goal Trade-off, changes to target amounts update the icons automatically.*

*Unlike the two-goal counterpart, users can change the target amounts by any value.*

### 4.3.3 Assets and Debt

Illustration 16 shows the Java prototype for the Assets and Debts interface. The panel is reachable from the "Assets" button on the top left and is the default panel when the tool opens, since the most common action will most like be updating savings on a monthly basis.


*Illustration 16: Software Interface, Assets and Debt*

*This sample profile shows all the main components that are visible in Concrete Budget's software implementation. Clicking on the top-left "Assets" button will show the panel on the right.*

The new Assets and Debt panel adds the "Current Savings" field, which was needed in order to simplify income distribution: now that we know how much a user has saved up, we can simply distribute this amount, without having to subtract expenses such as rent, utilities, credit card payments, etc. For the software, the savings field is actually the only one that causes a change in the sate of the system, since it is the amount that can be distributed to all other goals. The rest of the fields were added for future use and testing.

### 4.3.4 Goal Details

The goal details panel contains the most information and allows the most actions for any one goal. It contains information such as goal name, currently allocated savings amount, current milestone amount to reach, date of this milestone, overall target amount, minimum amount necessary, and the target date of this goal. The user can perform actions from this panel such as removing the goal completely, changing any of the fields, and comparing it with one or many goals.

As can be seen in Illustration 17, the name of the goal is presented in large font at the top along with the corresponding fuel gauge; the font color depends on whether this goal is on track or not, which reflects the color of the fuel gauge. The piggy bank icon is consistent with the Assets and Debt panel, guiding the user to the current savings information. The green gem is also a guide the users will quickly look for when looking for milestone information. Finally, the gold medal stands for the target goal which the user is aiming for; this area contains the information regarding the goal target.


*Illustration 17: Software Interface, Goal Details*

*This is an example of a goal details screen, which gives the user information regarding this goal and the ability to compare with one goal or many other goals in order to make this attainable.*

### 4.3.5 Goal Creation

Finally, we have the goal creation interface; although it was originally designed to be a pop-up, it was changed to simply be another panel, due to the negative view of users regarding pop-ups (thanks to spammers and phishers.) Illustration 18 shows how the panel format is still helpful to users and can quickly and easily be filled out.

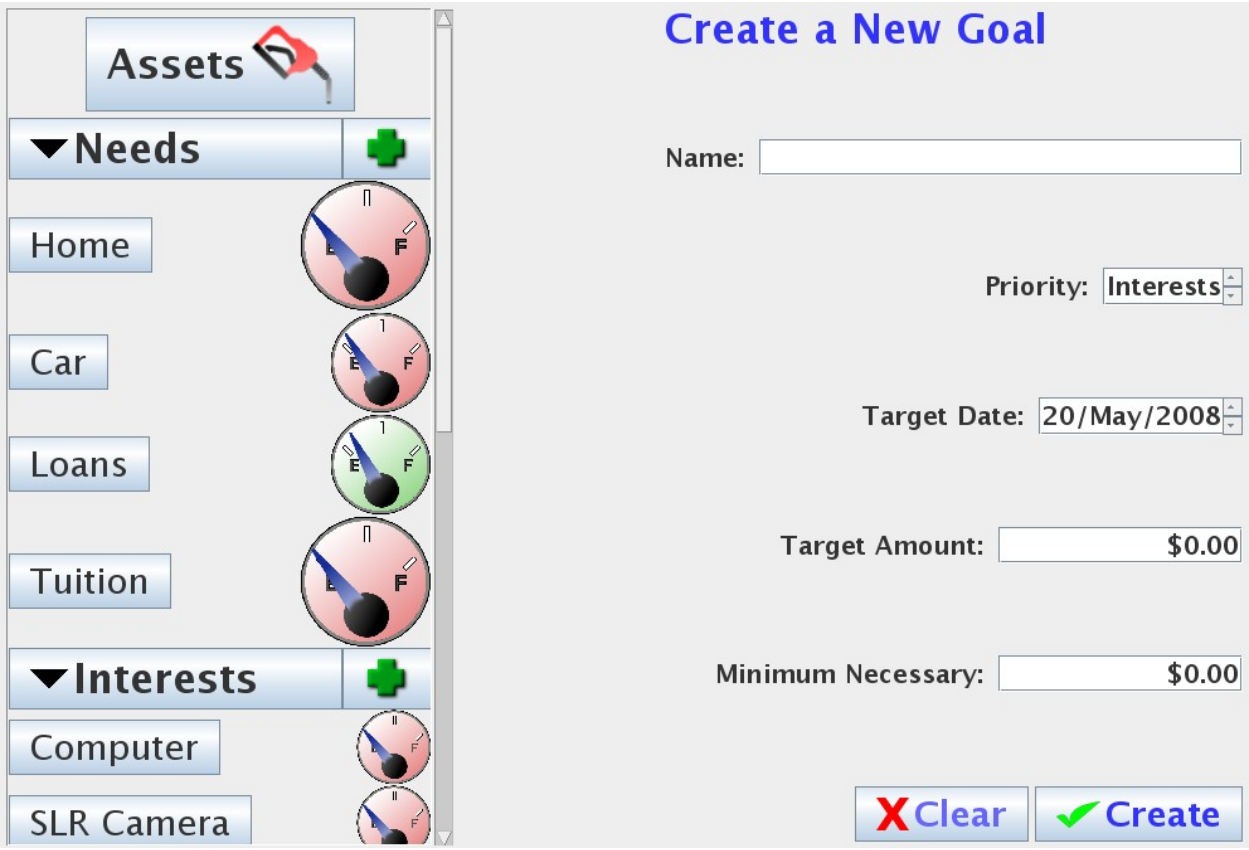


*Illustration 18: Software Interface, Goal Creation*

*This is the panel users would have to interact with in order to create a new goal. It only requires five fields, which is quick and easy to fill out – the goal of this panel.*

Once the user enters the appropriate information and presses the "Create" button, the software will perform some calculations, and display the goal details panel for the newly added goal. The goal will be added to the respective priority group and will show up on the left-hand side as soon as it is created. On the other hand, if the user clicks on the "Clear" button, it will simply erase all the content the user has inputed, and will start off from a clean slate.

## *4.4 Implementation of Software*

The following section is only meant for the tech-savvy types that are not afraid to five thousand lines of code in half a year. Because some of the classes are trivial and some were not even used (but could be used in the future,) not all classes will be discussed in this thesis.

The software was implemented in Java 6.0 using Eclipse with the exception of some auto-generated code from NetBeans for some of the complicated layout, such as GroupLayout which was mainly intended for Integrated Development Environments (IDEs), not manual coding. The Model-View-Controller design pattern (http://java.sun.com/blueprints/patterns/MVC.html) was followed to some extent – the model is clearly defined, but the view and the controller are a little less disjoint. One could say Main.java is the controller, but that would not be 100% correct. All classes in the UserInterface folder are meant to interact with each other, with View-Controller relationships.

### 4.4.1 Model Component

The model is essentially the back end of the system – anything the user simply gives input for, but never actually interacts with. In our case, it is also a way to simplify the gathering and passing of data. The classes used for this part are FuelGauge, Goal, GoalList, Pair, Person, and Profile.

The FuelGauge class was built in order to create a fuel gauge icon that has different color

background, three different sizes, and a needle that acts similar to the filled part of a progress bar. This is a very simple class that takes in a percent filled (the amount allocated divided by the target amount), which ranges from 0.0 to 1.0, the background color of the gauge, and the size of the gauge. This class is meant to be used in a static way, by calling createImage(percent, size, color).

The Goal class holds everything that is relevant to any one goal. It has many get and set methods which are particularly useful since this is where all of the data will be collected for goals. There are three helper methods (makeNeed, makeInterest, and makeWish) which are accessed in a static way and return an instance of a new goal with the inputs given (name, target amount, minimum amount, and target date.). There are also helper methods which generate some predefined goals in order to generate a list of goals for testing; these can be accessed in a static way by calling makeSampleNeeds, makeSampleInterests, and makeSampleWishes.

The Goal class also has a very helpful method that was used over and over which helps with the creation of ImageIcon's from the images folder. The method, createImageIcon, is called in a static way and takes in the filename of the icon in the images folder (NOTE: this is specifically hard-coded to be in the image folder – DO NOT USE for icons in a different folder.)

The GoalList class is simply an extension of Java's ArrayList which adds the functionality of moving a goal from its current position in the list to a new location. Currently, this is possible in two steps, but in an attempt to make a system that would be easy to develop for in the event of

drag-and-drop functionality, it was decided that a new class would be created with this added feature.

## Savings Distribution Algorithm

Profile is one of the most important classes along with Goal since it holds a lot of the user's information and performs the savings distribution to all the goals. It uses the distributeSavings method in order to distribute the savings any time something is changed in the system and uses two helper methods, distributeToMilestonesIfNotEnoughInSavings and distributeToTargetsIfNotEnoughInSavings. The way the savings distribution works is by first going through the user's needs and attempting to satisfy all current milestones. If there is enough money for all milestones in the needs, it simply distributes the money and goes on to the next step, otherwise it divides the savings amount by the number of goals in the needs GoalList and attempts to allocate them evenly to all goals – if any goal needs less than that amount, it is only given what is necessary, and the algorithm will loop around and allocate more money in list order to the goals. If there was enough to cover the milestones in the needs, it goes on to the interests and does the same with the remaining amount to be allocated. Should there be any money left over after this, it will go through the wishes and distribute it in the same way. Finally, if there is more money to be distributed, it will go back and attempt to fill the target amounts of each of the GoalLists, starting with the needs and ending with the wishes.

### 4.4.2 View-Controller Component

This is the more complicated section of the code, but rightfully so, since it is the main focus of this thesis. Some of the implementation details will not be mentioned due to the complicated nature of Java Layout Managers; for more information on how the content was layed out, please see the code, which is very well documented.

The AssetsAndGoalsPanel, seen in Illustration 19, extends Swing's JPanel, and is the only class that creates a panel for the left-hand side of the main window of Concrete Budget. This class takes in a Main object (discussed later in this section) and lays out all the goals in their corresponding priorities. It also creates the Assets button which calls on the Main object to change the right hand panel to the Assets and Debt panel. The AssetsAndGoalsPanel also deals with the plus buttons to create new goals and has a nice feature of allowing the goal lists to be collapsible (i.e. if one clicks on the priority name, it will collapse that group.)

The AssetsDebtPanel, seen in Illustration 20, is the panel that gathers all the information about the user regarding his or her assets and debt (e.g. credit card debt, current savings). The NewGoalPanel, seen in Illustration 21, has almost the same functionality as the AssetsDebtPanel, except it also creates new goals when the user clicks on the create button. When the create button is clicked on, the NewGoalPanel calls on its reference to the Main object in order to show the goal details of the newly created goal.

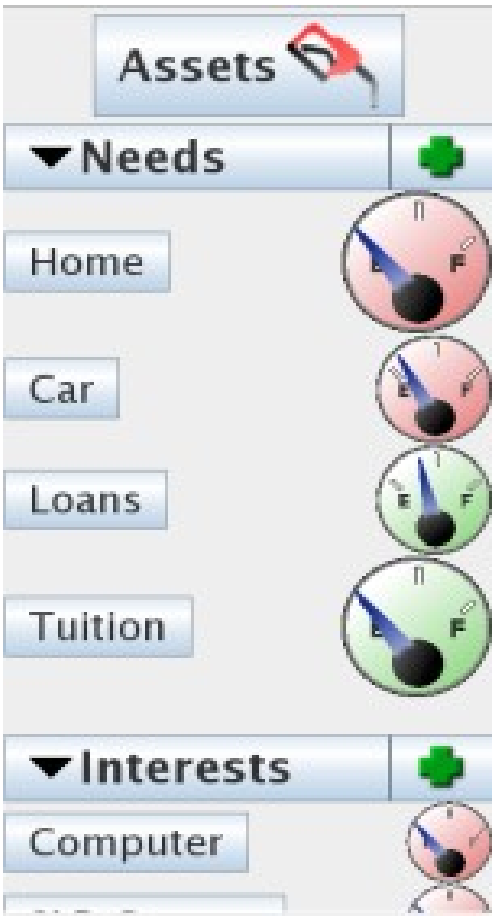


*Illustration 19: AssetsAndGoalsPanel*

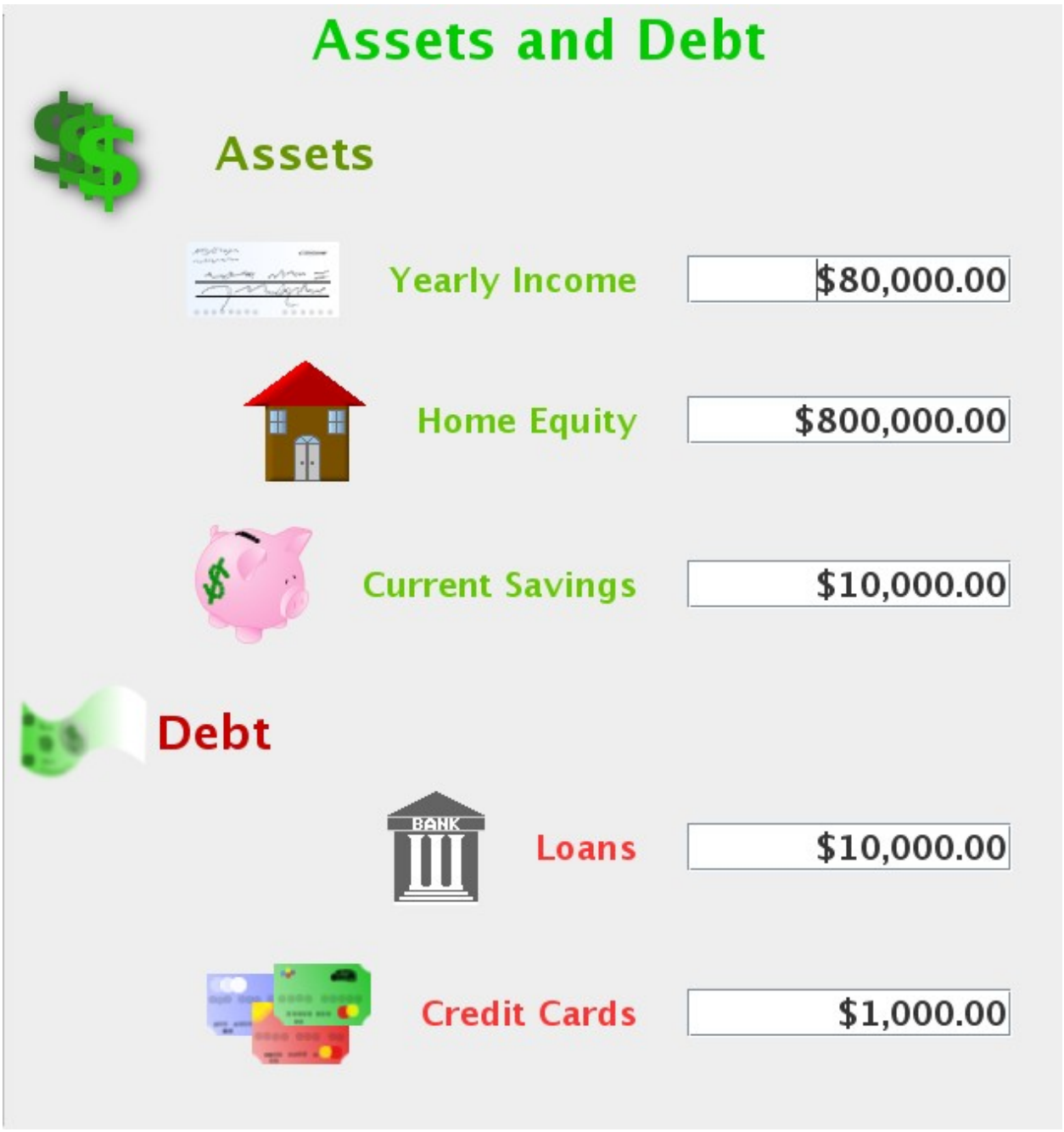


*Illustration 20: AssetsDebtPanel*


*Illustration 21: NewGoalPanel*

The GoalDetailsPanel, seen in Illustration 22, gives the details of a specific goal. It has a remove button which must remove the goal from the profile/GoalList and then will put the user in the

MultipleGoalsTradeOffPanel. The details panel also has a drop-down which allows users to trade off with one other goal, taking the user to the TwoGoalTradeOffPanel or clicking on the button next to the drop-down, which will also take the user to the MultipleGoalsTradeOffPanel.

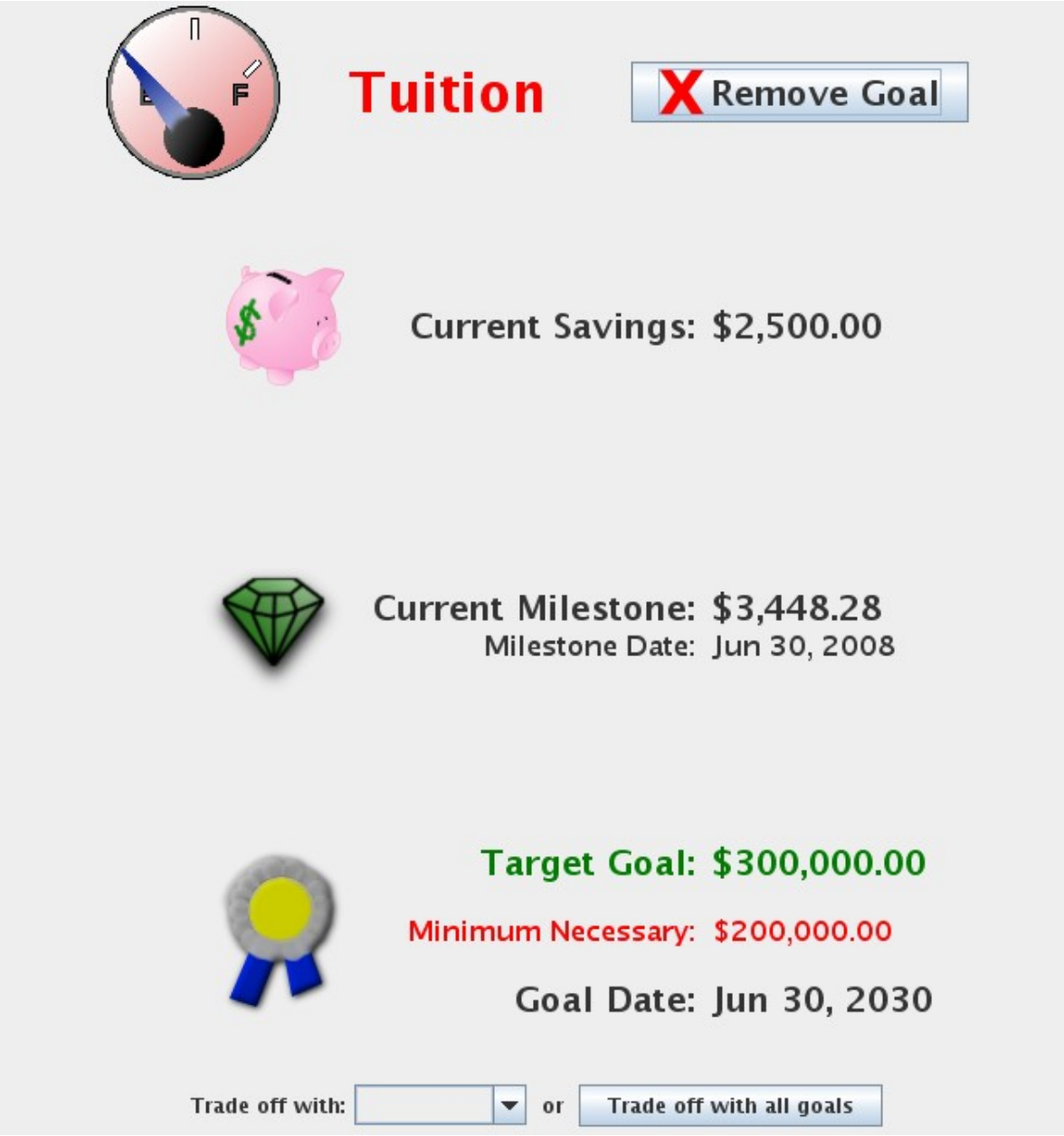


*Illustration 22: GoalDetailsPanel*

The MultipleGoalsTradeOffPanel (see Illustration 23) uses many GoalPanel's (see Illustration 24) in order to make a grid of goals which can be modified via spinners; this only modifies the target amount of a goal. The GoalPanel is simply a label with the name of the goal, the spinner showing the current target amount of the goal, and the fuel gauge that corresponds to that goal. The TwoGoalTradeOffPanel (see Illustration 25) shows the two goals, with their respective names and fuel gauges, and has a slider that goes from one side to the other, allowing users to swap amounts from one goal to the other. Below the slider, there are two text fields with the current target amounts for each goal, and a reset button, which returns the amounts to the value that was set before the TwoGoalTradeOffPanel was created.

Finally, we have Main, the most important class; it is the umbrella class which controls all other classes and is in charge of the entire frame (not surprising, it extends JFrame.) Main holds a Profile object, and two components, the left scroll pane and the right scroll pane. The left always contains a AssetsAndGoalsPanel, but the right can contain a set of different panels. In order for a class to ask Main to change the right panel, it must make a call to changeRightPanel with at least one other input: RightPanelState, which is a required input, and two other goals, which are optional depending on what one is attempting to do. The valid RightPanelState's are:

- *ASSETS*
- *NEW_GOAL*
- *DETAILS*
- *TWO_GOAL_TRADEOFF*
- *GOALS_TADEOFF*

By passing in one of these states and either null or optional goals, Main will change the right panel. And that's it. That is ConcreteBudget. Enjoy it!


*Illustration 23: MultipleGoalsTradeOffPanel*


*Illustration 24: GoalPanel*


*Illustration 25: TwoGoalTradeOffPanel*

# 5 Conclusion

In an age of recession and economic instability, where people simply pick a random number and say “that's how much I need to save in order to retire,” Concrete Budget is a must-have tool. Every person earning some form of income can benefit from a system that one defines with one's goals and that teaches one how to distribute one's savings in order to reach future goals. Although Concrete Budget is still in an early stage of development, it is clear that at some point in the near future, it simply makes sense to use a tool like this in order to not only stay out of debt, but manage one's life and learn how best to save money.

## *5.1 Future Work*

There are many aspects of Concrete Budget that could be made better, starting with conducting user testing in order to learn from and help the user. Adding the graph in Illustration 11 to the GoalDetailsPanel would be a great start. Determining how much a user should save per month, would also make economic sense, since right now users would have to add up all the milestones and subtract them from last month. Making use of better layout managers in most of the panels would help, since some of the icons and fields are not correctly aligned. Finally, adding the warning message on the bottom left of the wireframe prototypes would help users find how to trade off between goals, since it is currently somewhat hidden in the GoalDetailsPanel..This tool needs to be driven by the user, so listen to them and they will lead to a hugely successful program.

Risk is a very important notion in economics, but is much less tangible in people's minds because there is no easy way to calculate risk. With Concrete Budget's way of analyzing risk (asking for the minimum necessary, instead of a much more complicated risk calculation, and asking for the date of the goal), this information could be used in the back end in order to calculate or normalize the specific user's risk. This kind of information could be extremely useful to economists and investment portfolio managers.

Looking towards the future of financial planning, Concrete Budget could also be connected to the back end of a user's investment portfolio in order to hedge risks and automatically change a portfolio's distribution. Although this could all be done automatically, it should not be done so by default, as this is something very personal that everyone should decide with adequate information. As a parallel, the goals established by the user could also be presented to the person in charge of the family's investments in order to have a better understanding of how best to distribute the portfolio.

In terms of user testing, there would first need to be a brainstorming session with all possible tasks that users would want to perform in order to plan their budget. With theses tasks, and some work to add any missing functionality, one could set up a user study that would ask users to perform those tasks and rate the speed, ease, and satisfaction with which they accomplished the goal. Once this data is collected, the relevant information would be gathered and sent to the developers in order to improve upon the tool so as to make the user experience be as enjoyable as possible.

# 6 References


1. Find out where you are on the Boulevard to Retirement. Retrieved December 11, 2007, from Boulevard R.com - Your 5-Step Process to Get on Track for a Meaningful Retirement Web site: http://www.boulevardr.com/br/landing/home.jsf
2. Nielsen, J (2005). Ten usability heuristics. Retrieved December 11, 2007, from Heuristics for user interface design Web site: http://www.useit.com/papers/heuristic/heuristic_list.html
3. MoneyWell. Retrieved December 11, 2007, from No Thirst Software - MoneyWell Web site: http://nothirst.com/moneywell/
4. myPlan Snapshot. Retrieved December 11, 2007, from Fidelity Investments: myPlan Snapshot Web site: http://personal.fidelity.com/planning/retirement/content/myPlan/index.shtml
5. Olsen, Henrik. "Balancing fidelity in prototyping." GUUUI - Balancing fidelity in prototyping. GUUUI. 20 May 2008 <http://www.guuui.com/issues/03_05.php>.
6. Royce, Winston (1970), "Managing the Development of Large Software Systems", Proceedings of IEEE WESCON 26 (August): 1-9, http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf.
7. “Paper Prototyping.” Snyder Consulting - Paper Prototyping. Snyder Consulting. 20 May 2008 <http://www.snyderconsulting.net/article_paperprototyping.htm>.