1. INTRODUCTION

What is Software Design?

Before we answer that question, let's look at an important sub question - What is Design - And why is it so important?

To answer in a single sentence - "so that things don't happen by accident."

Design is an important element of innovation that is often overlooked. It helps determine how we interact with, and experience, products and services and this, in turn, will affect which products or services we will buy and what we are prepared to pay for them. Good quality design is an integral part of sustainable development. Achieving good design is about creating places, buildings, or spaces that work well for everyone, look good, last well, and will adapt to the needs of future generations.

Good design responds in a practical and creative way to both the function and identity of a place. It puts land, water, drainage, energy, community, economic, infrastructure and other such resources to the best possible use – over the long as well as the short term. It differentiates a product or service, turning insights into customer preference. Properly applied, design can give you a sustainable advantage, help you command a premium price, gain market share and even reduce production costs.

So how do we introduce and incorporate all these features of Design in the field of Technology? Enter Software Design.

Wikipedia defines Software Design as the process by which an agent creates a specification of software artifact, intended to accomplish goals, using a set of primitive components and subject to constraints.

In other words, a software design creates meaningful engineering representations (or models) of some software product that is to be built.

Software plays very crucial role in all activities such as banking, finance, transportation, hospitality, medical, law, business, industries, government etc.

Almost every sector is dependent on software for its operation, thus evolving to the field of software engineering, which in turn has led to the growth of software design. All these different industries have one thing in common - they all require a good design that lasts a long time and is stable and easily upgraded in the future. Needless to say, before the advent of Software design, keeping track of all the needs and requirements of a steadily growing industry was a humungous task.

2. PURPOSE

When we start a software project from scratch, our first step should be deciding the design of the project. Design will not only allow us to visualize how the product is going to turn out, but it'll also allow both software owners and developers to realize how it's going to function.

Let us take an example - You're required to write some code for an every-day task you do.

A good programmer says 'I can solve this problem' and writes 100 lines of code. A great programmer says 'I've seen this problem before' and re-uses the code he/she wrote last time.

Designing a program is sitting down, thinking about the various issues of the problem-domain and working out in a general fashion how you might solve that issue, or how you've solved it before, and when to decide what's important and what's not.

It is therefore important for a software design to be functional, simple and readily informative. This way, even a child could readily use your application without taking much time to learn it's functionality. That's what makes up an intuitive app design. That's where you can start from.

A good software project, must inculcate all the qualities of good design. To help us measure the correctness of our design, we have the Software Design Principles.

The design principles are associated to Robert Martin who gathered them in "Agile Software Development: Principles, Patterns, and Practices". According to Robert Martin there are three important characteristics of a bad design that should be avoided:

- Rigidity - It is hard to change because every change affects too many other parts of the system.
- Fragility - When you make a change, unexpected parts of the system break result is you have to do impact analysis in whole project.
- Immobility - It is hard to reuse in another application because it cannot be extricate from the current application.

As a general rule of thumb, we need to keep in mind that a good software design is an iterative process. We must always get feedback on our design and find ways to simplify and make the design fluid, without affecting the final outcome.

A good design leads to software that is:

- Correct – does what it should
- Robust – tolerant of misuse, e.g. faulty data
- Flexible – adaptable to shifting requirements
- Reusable – cut production costs for code
- Efficient – good use of processor and memory

Like Antoine de Saint-Exupery once said - "You know you've achieved perfection in design, not when you have nothing more to add, but when you have nothing more to take away."

3. PROJECT DESCRIPTION

Now that we have a clear idea of what Software Design is and how important it is to us, we can start looking at the details of our project.

The project is mainly divided into 2 deliverables:

The first deliverable deals with the design of an industry level software that can take input from the user, calculate some values that could maximize the dimensions of a cube, to be made from a given piece of wood, and present the results in a flexible visual format.

By the end of the first deliverable, the software had to satisfy the following objectives:

- The software should calculate the maximum cube dimensions that can be made from one long piece of wood.
- The software should give the user the option to enter the size of the wood in metric units (centimeters/meters) or American units (inches/yards).
- The software should calculate the maximum possible cube that could be made with the given piece of wood.
- It should display the cube's side length as well as it’s volume in text form.
- The software should also display a drawing of the resultant cube, showing the dimensions of the cube side as well as the volume.
- The user should be able to switch the input as well as the output values between the two measurement systems at will. The program must offer flexibility of transforming between the two standards (American and Metric) for both the input and the output.

The second deliverable deals with improvising on the first deliverable. Along with the cube, we are supposed to maximize the dimensions of a two more geometric shapes - a rectangular-based pyramid and a triangular based pyramid, that can be made from a given piece of wood, and present the results in a flexible visual format.

The final, end product is a an industry level software which combines the objective of both deliverable one and two, and is user-intuitive, easy to operate and most importantly, has a good software design.

In the following sections, each and every aspect of this project and how we solved each objective is explained in detail.

4. PROTOTYPES

A prototype is an early sample, model, or release of a product built to test a concept or process or to act as a thing to be replicated or learned from. A prototype is designed to test and try a new design to enhance precision by system analysts and users. Prototyping serves to provide specifications for a real, working system rather than a theoretical one.

4.1 CHARACTERISTICS AND LIMITATIONS OF PROTOTYPE:

Engineers and prototyping specialists seek to understand the limitations of prototypes to exactly simulate the characteristics of their intended design.

It is important to realize that by their very definition, prototypes will represent some compromise from the final production design. Due to differences in materials, processes and design fidelity, it is possible that a prototype may fail to perform acceptably whereas the production design may have been sound. A counter-intuitive idea is that prototypes may actually perform acceptably whereas the production design may be flawed since prototyping materials and processes may occasionally outperform their production counterparts.

In general, it can be expected that individual prototype costs will be substantially greater than the final production costs due to inefficiencies in materials and processes. Prototypes are also used to revise the design for the purposes of reducing costs through optimization and refinement.

4.2 DIFFERENT TYPES OF PROTOTYPE:

4.2.1 PAPER PROTOTYPES

In human–computer interaction, paper prototyping is a widely used method in the user-centered design process, a process that helps developers to create software that meets the user's expectations and needs—in this case, especially for designing and testing user interfaces. It is throwaway prototyping and involves creating rough, even hand-sketched, drawings of an interface to use as prototypes, or models, of a design.

While paper prototyping seems simple, this method of usability testing can provide a great deal of useful feedback which will result in the design of better products. This is supported by many usability professionals.

4.2.2 DIGITAL PROTOTYPES

Digital Prototyping gives conceptual design, engineering, manufacturing, and sales and marketing departments the ability to virtually explore a complete product before it's built. Industrial designers, manufacturers, and engineers use Digital Prototyping to design, iterate, optimize, validate, and visualize their products digitally throughout the product development process.

Innovative digital prototypes can be created via CAutoD through intelligent and near-optimal iterations, meeting multiple design objectives (such as maximized output, energy efficiency, highest speed and cost-effectiveness), identifying multiple figures of merit, and reducing development gearing and time-to-market.

Marketers also use Digital Prototyping to create photorealistic renderings and animations of products prior to manufacturing. Companies often adopt Digital Prototyping with the goal of improving communication between product development stakeholders, getting products to market faster, and facilitating product innovation.

4.3 PROTOTYPE FOR DIFFERENT GEOMETRIC FIGURES

4.3.1 CUBE - PAPER PROTOTYPE:

a) We were given a piece of wood and had to make a cube out of it.

b) Only the side length of the wood was considered. The height and the breadth were ignored.

c) Divided the length of the wood into 12 equal parts. Labeled the length of each part as side length.

d) Applied the formula Volume = side length * side length * side length.

4.3.2 CUBE - WOOD PROTOTYPE:

a) We received a wood of 12m length.

b) The wood was cut into 12 equal parts.

c) Attached all the sides as per the paper prototype.

d) The height and breadth of cube was not considered, hence the actual volume was different to the one which was calculated through the formula which was calculated during paper prototyping.

4.3.3 CUBE - DIGITAL PROTOTYPE:

a) It was designed based on the paper prototype.

b) Since it was based on the paper prototype we did not consider the height and breadth.

c) The user enters the length of the wood.

d) The user can enter the input and output metrics.

e) The volume is calculated based on the formula = innerside * innerside * innerside.

f) The same formula is applied for inner side except it being (side-breadth).

4.3.4 RECTANGULAR PYRAMID - PAPER PROTOTYPE :

a) We were given a piece of wood and had to make a cube out of it.

b) Only the side length of the wood was considered. The height and the breadth were ignored.

c) To maximize the volume we thought of considering a square based pyramid.

d) Divided the length of the wood into 8 equal parts. Labeled the length of each part as side length.

e) Applied the formula Volume = (side* side * side)/ (3*1.1414).

4.3.5 RECTANGULAR PYRAMID - DIGITAL PROTOTYPE**:**

a) It was designed based on the paper prototype.

b) Since it was based on the paper prototype we did not consider the height and breadth.

c) The user enters the length of the wood.

d) The user can enter the input and output metrics.

e) The volume is calculated based on the formula = (side* side * side)/(3*1.1414).

f) The same formula is applied for inner side except it being (side-breadth).

4.3.6 TRIANGULAR PYRAMID - PAPER PROTOTYPE:

a) We were given a piece of wood and had to make a cube out of it.

b) Only the side length of the wood was considered. The height and the breadth were ignored.

c) Divided the length of the wood into 6 equal parts. Labeled the length of each part as side length.

d) Applied the formula Volume = length/ 8.32.

4.3.7 TRIANGULAR PYRAMID - WOOD PROTOTYPE:

a) We received a wood of 12m length.

b) The wood was cut into 6 equal parts at an angle of 30® on both ends on the side.

c) Attached all the sides as per the paper prototype.

d) Since we did not consider the height and breadth of the wood, the actual volume was different to the one we calculated through the formula formulated during the paper prototyping.

4.3.8 TRIANGULAR PYRAMID - DIGITAL PROTOTYPE:

a) It was designed based on the paper prototype.

b) Since it was based on the paper prototype we did not consider the height and breadth.

c) The user enters the length of the wood.

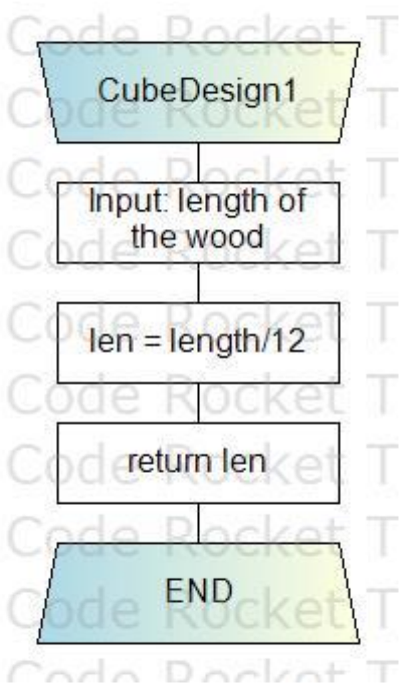
d) The user can enter the input and output metrics.

e) The volume is calculated based on the formula = (side* side * side) / (6*1.1414)

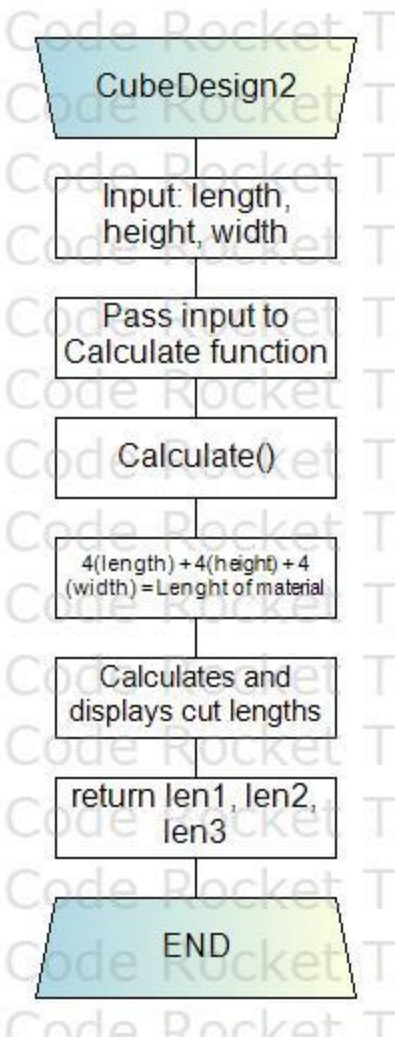
f) The same formula is applied for inner side except it being (side-breadth).

5. FLOWCHARTS

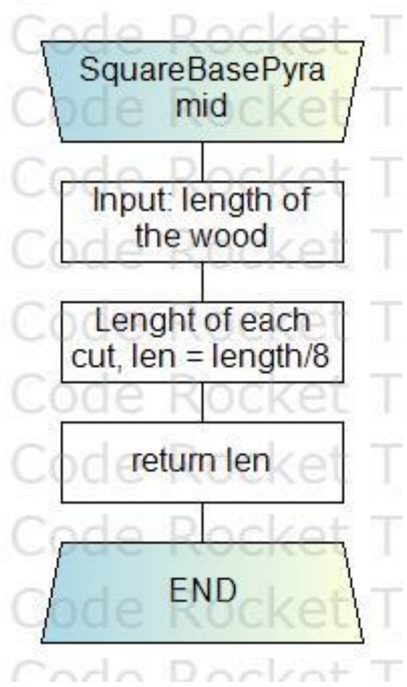
CUBE: DESIGN 1



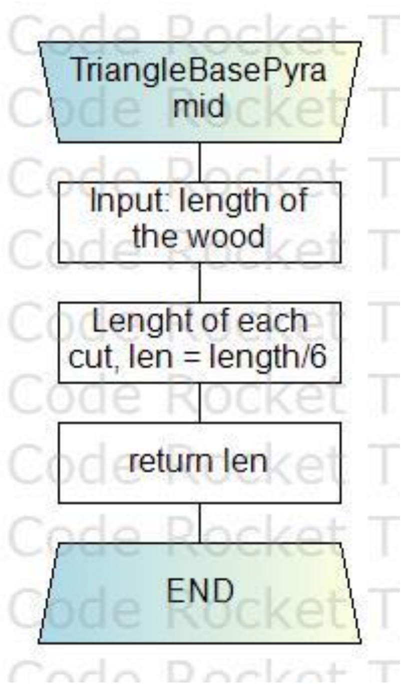
CUBE: DESIGN 2


CubeDesign2
Input: length, height, width
Pass input to Calculate function
Calculate()
4(length) + 4(height) + 4 (width) = Lenght of material
Calculates and displays cut lengths
return len1, len2, len3
END

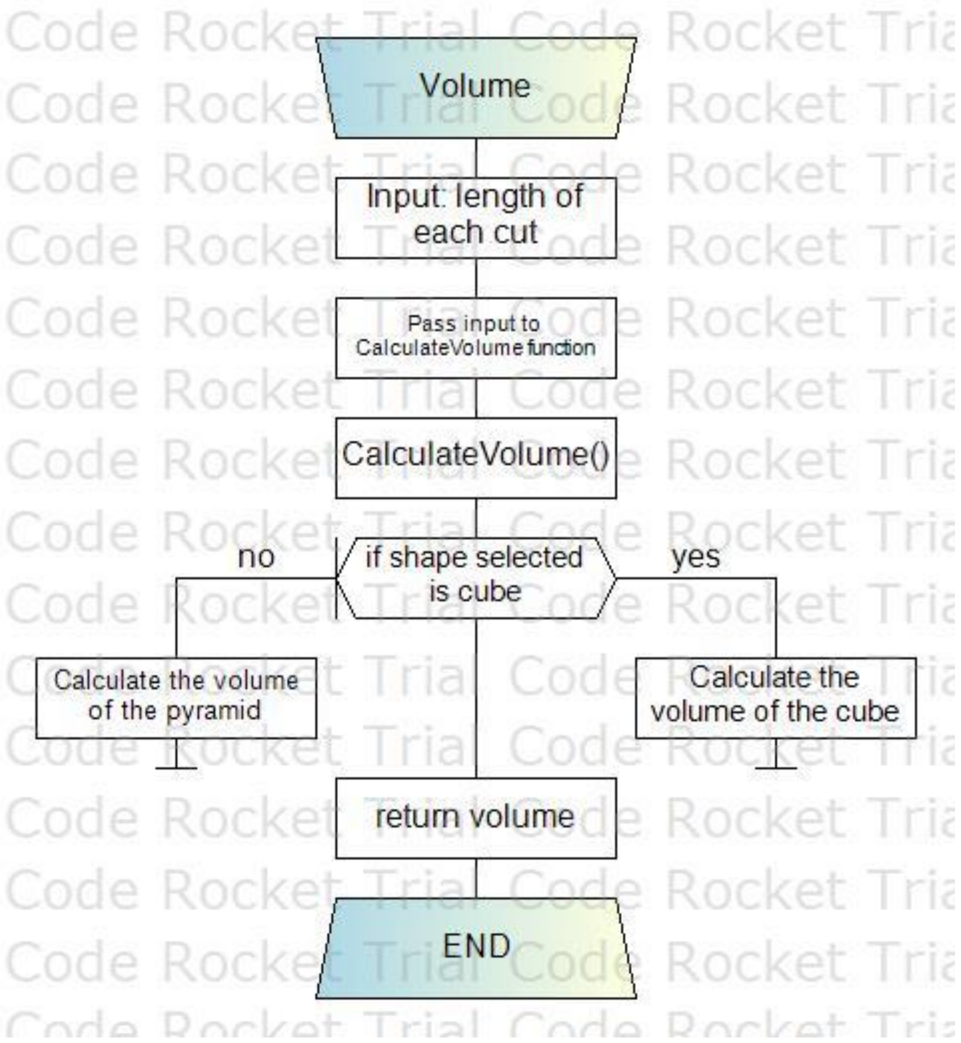
RECTANGLE-BASED PYRAMID



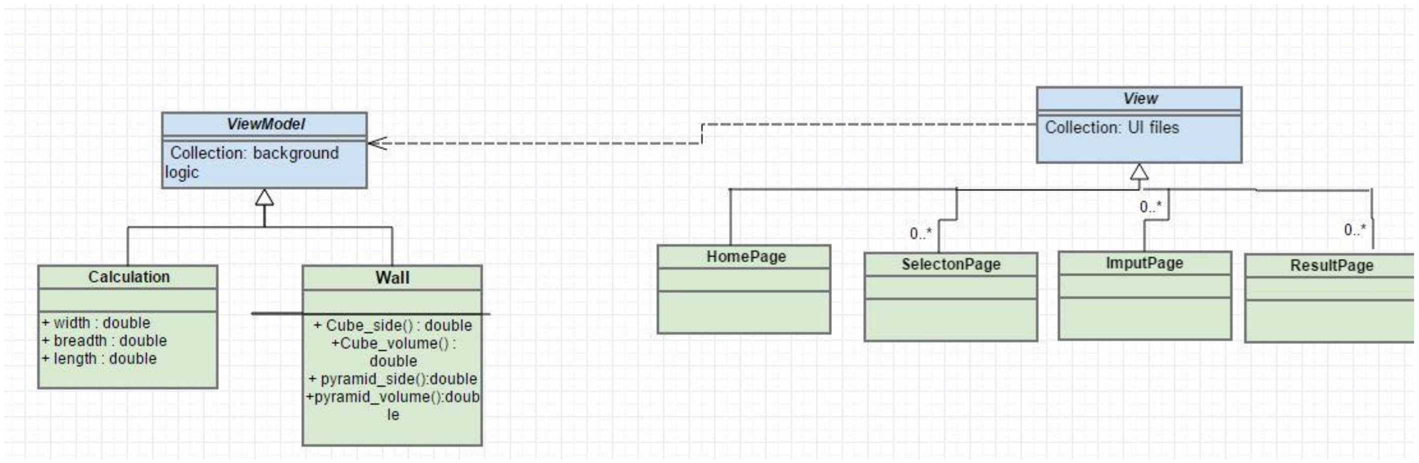
TRIANGLE BASED PYRAMID



CALCULATING VOLUME


Volume
Input: length of each cut
Pass input to CalculateVolume function
CalculateVolume()
if shape selected is cube
no
yes
Calculate the volume of the pyramid
Calculate the volume of the cube
return volume
END
Code Rocket Trial

6. SYSTEM ARCHITECTURE



The above is a schematic depiction of the source-code architecture. This diagram conveys some general principles in the form of a UML Class Diagram. The package View comprises of all the classes and methods required to display the Graphical User Interface of the software, while the package ViewModel consists of the classes performing background logic whereby the necessary computations are performed to generate the required figures.

7. TEAM EXPERIENCE

The overall experience as a team has been a really pleasant one. To accomplish some purpose through working among a group can often be challenging because numerous members have to effectively agree and cooperate. But to overcome those challenges and arrive at a solution collectively is the whole notion of working in a team. Having worked on the project for 4 months, we have all come to understand what each of us brings to the table. It's amazing how with just one brainstorming session, we can start looking at things in a whole new perspective.

We all met on the first day of the course Software Design. It was a pleasant surprise how with just 8 people, we had so much of professional experience among ourselves. While some had experience in coding, product development and deployment from their work days in IT companies, others offered innumerable amount of insight into topics they'd previously studied on in different courses.

As soon as we were given our first deliverable, we got to work. The whole team was committed throughout the whole. The team met several times a week, which was very productive. These meetings went well, but could have gone better if we had a more structured schedule to maximize our time together.

Although the first task seemed rather trivial, when we started discussing our strategies, it proved to be a rather tedious job. Our first deliverable involved finding out a way to design a cube from a given piece of wood and to maximize the output. We successfully worked out the details of this deliverable and were quite content with ourselves for our progress.

We were asked to not only design an electronic prototype of the cube, but also a physical one too. Working with a real piece of wood, without the ability to "undo" our actions while working out the details, we were at first taken over by irritation which delayed our completion of the prototype. But we took it positively and thought of this as a chance to learn the effects of bad software design in a hands-on environment.

Having learnt from our mistakes and with a new sense of clarity, we modified our prototype to the norms of the given objective. Similar to task 1, we had another go at creating a physical prototype - when we were asked to design a triangular based pyramid. This time we handled the objective a lot more effectively. This helped us make qualitative changes to our software product. In the midst of these two main objectives, we also had chances to collaborate when we were given a couple of reading assignments and also a topic - User Centered Design - to present.

In the end, we have managed to succeed together because not one of us had the ego of wanting to be the best on the team, but instead all wanted our team as a whole to be the best. We constantly had energetic group meetings, which meant we were communicating well with each other. The team was strong from the beginning through the development stages, even though we might have had some disagreements.

Everyone agreed that every member was reliable and had contributed to the group role important in progressing collectively. Our group thrived off of the synergy composed by coming together as a team.

8. CONCLUSION

The design of software has been a focus of software engineering research since the field's beginning. Software Design will continue to remain a principal focus. After multiple rounds of registering feedbacks for the prototypes of our software, we have managed to complete all the objectives of the project. But just because we have finished satisfying the objectives doesn't mean our software is capable of performing well forever. We always need to remember that software design is an iterative process and there is always scope for development and improvement.

9. REFERENCES

http://www.artima.com/weblogs/viewpost.jsp?thread=331531
http://www.clubofamsterdam.com/content.asp?contentid=545
http://www.cs.wustl.edu/~levine/courses/cs342/c++/design- principles_4.pdf
http://www.nada.kth.se/~karlm/prutt05/lectures/prutt05_lec7.pdf
https://avilay.wordpress.com/2012/09/01/what-is-good-software-design/
http://mobidev.biz/blog/importance_of_design_in_software_development
http://www.liquidplanner.com/blog/designing-better-software-5-rules-to-follow/
http://www.ics.uci.edu/~andre/informatics223s2007/taylorvanderhoek.pdf