SFWR ENG 3A04: Software Design III - Large System Design

Dr. Ridha Khedri

Department of Computing and Software, McMaster University
Canada L8S 4L7, Hamilton, Ontario

1. Preamble
2. What is Design?
   - Intuitive Understanding of Design
   - Exploring Aspects of design
   - An understanding of design

- In biology, there is a law called the law of **least effort**
- It postulates that living organisms prize energy
- It is usually hard to come up with energy and it is exigent to extract it from the environment

- For survival organisms take every opportunity to store energy and strive to use as little of it as possible
- Many examples in biology show that energy is difficult to obtain, and even when obtained, it is costly to store and very easy to lose
- To survive, organisms always strive to reduce as much as possible the energy cost of their activities all through their lives

In engineering,

- We aim at having systems that
  - can resist the test of time and,
  - all through their lives, use as little energy as possible
- We ought to design them so that they need the least effort for building, assessing, operating, and maintaining

- Resources such as money (Money is an expression or a form of energy that we use to sustain our life), as energy for organisms, are extremely scarce and valuable
- Hence, all measures have to be deployed to require as little of them as possible to sustain the development and the live of an engineering artefact
- If the building of a system demands less resources (energy) and its assessment is expensive, then we can say that the system is not abiding by the law of least effort
- Who tried to test a badly designed program knows how time-consuming building an appropriate test suite for a white-box testing

Example

We are to design a program that calculates the sum of the first 100 natural numbers staring from 1. Let S be this sum. Hence, $S = 1+2+3+4+\cdots+98+99+100$.

- A designer noticed that the sum can be accumulated by starting from S holding the neutral element of the addition, which is 0, and then adopting a repetitive behaviour to add the numbers going from 1 until reaching 100.

```
int naive_sum(int n){
    int sum = 0;
  for (int i=1; i <= n; i++)
    {
        sum += i;

    }
    return sum;
}

int main()
{
  int count;
  cin >> count;
  naive_sum(count);
  return 0;
}
```

Is there another design?

- That of Carl Friedrich Gauss (April 30, 1777 – February 23, 1855)
- He is a German mathematician who is regarded as the best mathematician. He made significant contributions to several areas in mathematics and physics such as number theory, geometry, probability theory, geodesy, planetary astronomy, the theory of functions, and potential theory. It is known about him that he was a calculating prodigy, and he developed the ability to do elaborate mental calculations.

Carl was asked to calculate the sum of the natural numbers between 1 and 100

- Carl noticed that if you add the first number 1 and the last number 100, you get 101
- It is the same result for the second first number, 2, and the second last number, 99
- The result is also the same for the 50 pairs $(1 + m, 100 - m)$, for $m \in [0, 49]$
- Let us call n the number of numbers from 1 to 100 that we need to add (i.e., $n = 100$)
- Hence, for Carl, the sum $S = \frac{n(n+1)}{2}$

```
int sum(int n){
   return (n * (n + 1)) / 2;
}

int main()
{
  int count;
  cin >> count;
  sum(count);
  return 0;
}
```

- Which of the two designs is easier to write, assess (either by testing, proof of correctness, or inspection), and maintain?
- Which of the the two designs satisfies the principle of simplicity of mechanisms?

- The common practice in the software development of today is to focus only on the development stage
- What cost is incurred by the other stages of maintenance or testing (which very often omitted or partially carried) is deemed to be irrelevant to the developer
- One has to consider the cost for the whole life cycle
- Sustainability is related to this law of least effort
- With this current practice, the amount of resources that a system needs is just way too much for the environment to provide

- One interpretation of a system being in harmony with its environment: The system only needs as tittle resources as possible to provide the environment of what it expects from it
- Its existence will not be sustained, if it needs more than what the environment can provide
- The environment will scrap it or cast it aside, if it does not provide what it expects from it and that with the required qualities of performance, reliability, precision, etc.

- For the survival of a system in its environment,
  - its harmony in exchanging benefits is needed
  - its reliance on the lowest possible resources for its development, assessment, operation, and maintenance

> The design of systems that abide by the law of least effort and that are in harmony with their environments is the purpose of good designers

- This activity is a characteristic of human being
- While we find in nature many animals that build sophisticated constructions, they cannot be called designers
- For instance, ants and termites (mound-building termites) live in underground colonies or built inside trees, and they create their homes
- They build shelters, called mounds, that have a diameter of 30 metres (98 ft)
- They select the right environment for their mounds by locating them in well-drained areas

Termite Mounds in the Bungle Bungle Range in Western Australia

(By Ouderkraal - Own work, CC BY-SA 3.0,

https://commons.wikimedia.org/w/index.php?curid=17932901)

- This construction has a vey complex function that of a dependable shelter and a form that integrates well with the colony's environment
- Can we call termites designers?
- There is no evidence that, within a considered environment, termites change the way they build their mounds
- They do not design other artefacts
- They build their mounds by instinct: they build them naturally and without having to think or learn about constructing them

- It is different for human beings
  - From early ages of humanity, human built diverse tools
  - Stone tools are the most well-know residues of human of stone age, which began about 2.6 million years ago
  - They built them in different forms and shapes
  - These conceived artefacts that human conceived and built through the ages helped humanity to get stronger
  - Stone tools empowered human of the stone age to better use the meat of large animals, providing them of more energy and food to survive
  - At the same time the making the early stone tools required relatively a little efforts
  - Even, can be build on site without the need to lose energy for carrying them; a manifestation of the law of least effort

- One cannot define design as would define a mathematical activity such as “rewriting“, “derivation”, etc.
- The nature of design is not easy to grasp in a clear definition
- Looking for a definition of design may not help you grasp what it is
- Design can be seen in any human artefact and in nature
- The single word “design” encompasses an awful lot of objectives and subjective aspects

- They can be
  - aesthetic (subjective)
  - functional (Objective)
  - many other aspects of an object or a process (Objective or subjective), which usually requires considerable
    - research,
    - thought,
    - modeling,
    - interactive adjustment,
    - and re-design

- Depending on the designed entity, either the objective or the subjective aspects might takeover
- When the subjective aspects take over, design is then viewed as a more rigorous form of art, or art with a clearly defined purpose
- When the objective aspects take over, design is then viewed as a simple mathematical transformation of the mathematical entities expressing the functional requirements into a design which is a collection of mathematical entities with their mathematical connectors
- So, design involves both mathematics and art as transformation means (and may be other skills?!!!)

Example

Design as a mathematical activity Construction of a *deterministic finite automaton* (DFA) that accepts all binary strings with an even number of 0's and the number of 1's is a multiple of 3.

Design Process: Let $L(M_3)$ be the language that contains only all binary strings with an even number of 0's and the number of 1's is a multiple of 3.

$L(M_3)$

= ⟨ From the textual description of the languages as given by the question. ⟩

$\{x \mid x \in \{0,1\}^* \;\wedge\; x$ contains even number of 0's $\wedge\; x$ contains a number of 1's that is multiple of 3$\}$

= ⟨ Definition of set intersection ⟩

$\{x \mid x \in \{0,1\}^* \;\wedge\; x$ contains even number of 0's$\}$ $\cap \{x \mid x \in \{0,1\}^* \;\wedge\; x \;\wedge$ contains a number of 1's that is multiple of 3$\}$

Example

From the above derivation, we can notice that $L(M_3)$ is the intersection of two languages:

1. $L(M_1) = \{x \mid x \in \{0,1\}^* \;\wedge$ x contains even number of 0's$\}$
2. $L(M_2) = \{x \mid x \in \{0,1\}^* \;\wedge\; x \;\wedge$ contains a number of 1's that is multiple of 3$\}$

$L(M_1)$ and $L(M_2)$ are regular since they are accepted by the machine M_1 and M_2, respectively.
Since $L(M_3) = L(M_1) \cap L(M_2)$, then M_3 can be obtained from M_1 and M_2 using the product construction.

Example

Continued ...

Product construction:

1. $Q_3 = \{\alpha, \beta\} \times \{a, b, c\} = \{(\alpha, a), (\alpha, b), (\alpha, c), (\beta, a), (\beta, b), (\beta, c)\}$
2. $\Sigma = \{0, 1\}$
3. $s_3 = (s_1, s_2) = (\alpha, a)$
4. $F_3 = F_1 \times F_2 = \{\alpha\} \times \{a\} = \{(\alpha, a)\}$
5. δ_3 is given by the following table.

	0	1
(α, a)	(β, a)	(α, b)
(α, b)	(β, b)	(α, c)
(α, c)	(β, c)	(α, a)
(β, a)	(α, a)	(β, b)
(β, b)	(α, b)	(β, c)
(β, c)	(α, c)	(β, a)

Example

Continued ...

The graphical representation of M_3.

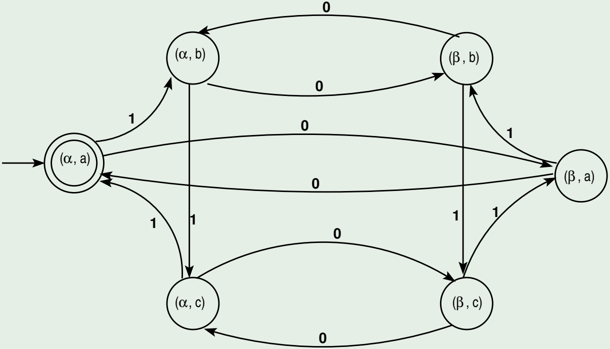

Figure: Machine M_3 (The product of M_1 and M_2)

Example

Design as an artistic activity

Figure: Game Interface

Example

Design as an artistic activity (Continued...)

In the literature, you find some unsuccessful attempts to define the term design:

- In some literature, you find that design is "making things better for people"
- But, for instance, we all could say that we too "make things better for people"!!!!!!!
- "Design is that area of human experience, skill and knowledge which is concerned with man's ability to mould his environment to suit his material and spiritual needs."
  [Archer, B (1973) The Need for Design Education. Royal College of Art]

- Design could be viewed as an activity that translates an idea/goal/wish into a blueprint for an artefact or a process that is fit for its environment
- We will focus on examples of design that would give each one of us his/her own understanding of design (which enhances creativity)
- We however agree that the obtained artefact or process is suitable for (fit for use in) the intended environment

What is in design more than the translation of an idea to an artefact or process?

- Design can be seen as an activity of reconciliation between conflicting needs or constraints put by the environment
- Design is a creative process based around the "building up" of ideas
  - Avoiding negative judgments of contributors when designing eliminates the fear of failure and encourages maximum input and participation
  - Encourage outside the box thinking in this process since this can often lead to creative solutions

- Although design is subject to personal taste, design thinkers share a common set of values that drive innovation
- These values are mainly creativity, fertile imagination, "whole brain thinking", teamwork, end-user focus, curiosity
- The design thinking process should evolve through the following stages: define, research, ideate, prototype, choose, implement, and learn

- There are many attempts to understand/define design
- They are all articulated around the following understanding:

Working Statement on Design

Design is the process of embedding a function in a form that is dependable, sustainable, and in harmony with its environment.

The Function

- The function is the fundamental reason for the existence of the designed object, whether it is a system, a process or any other artefact
- We need the designed object to accomplish a function
- In some areas of design this function could be the identity function (does not change its environment by transforming an input to an output)

- We will touch this kind of identity function when we discuss artistic design
- We will omit the design objects that do have an identity function
- We assume that our design objects have a strictly different from the identity function

- The function of a design object is to mould our environment by taking something from the environment and transforming it
- A computer system takes an input and gives to its environment an output
- The form of a design object has to be fit for its environment

- A bridge on a river that is intended to be for decades has to be bit for the geological attributes and climate attributes of its environment
- The designer should always remember that her design is for a specific environment
- Once the environment changes, the design has to be revised

The Form

- The form is about the material manifestation of the design object in its environment
- In other terms, it is the embodiment of the function delivering entity in its environment
- This embodiment could be physical, cyber, or other
- As a material object, it has to have attributes that tell about it (fast performing, secure, easy to use, etc.)
- These attributes should be able to tell us about how fit this object to its environment

Function and Form

- The function is fundamental (main purpose of existence $\Longrightarrow$ the function is independent from the form)
- The form is derived from the function
- The environment in which the design (function + Form) is going to be embedded
- The form has to be adjusted to the environment of the design object

Example

The Function:

- If we require to design a program that sorts a set of numbers
- If the designer returns the design of a program that does not sort, then the designer did not meet the main purpose of the design
- Whatever desired form that this delivered program has, it will not be acceptable as a design $\Longleftarrow$ the fundamental reason for the existence of the design object is not met

Example

Example continued ...

The Form:

- If the designer delivers a program that indeed sorts numbers, then we start to discuss the program's form
- Certainly this form has limits put by both the function and the environment
- In our case of a sorting program design, we know that its performance couldn't be better than $O(n \log n)$ in the best case, $O(n \log n)$ in the average case, and $O(n^2)$ in the worst case, which is the performance of Quicksort algorithm

Example

Example continued ...

Form:

- In this case, we notice that the function puts limits on one of attributes of the program's form
- The memory on the device on which the program is intended to run, which constitutes one aspect of the program's environment, might as well put a constraint on the number of numbers that can be sorted
- It is important to keep in mind that the form of a design object follows its function and has to fit and mould its intended environment

- The embedding of a function within a form is carried through a process that we refer to as the design process
- The **output** of the design process is the specification of the most viable design solutions
- The design process starts from a specification of a design problem (Requirements specification)
- In the next lecture, we will recall:
  - How to determine the requirements on the function?
  - How to determine the requirements on the form?
  - How to determine the specific aspects of the environment?

The End