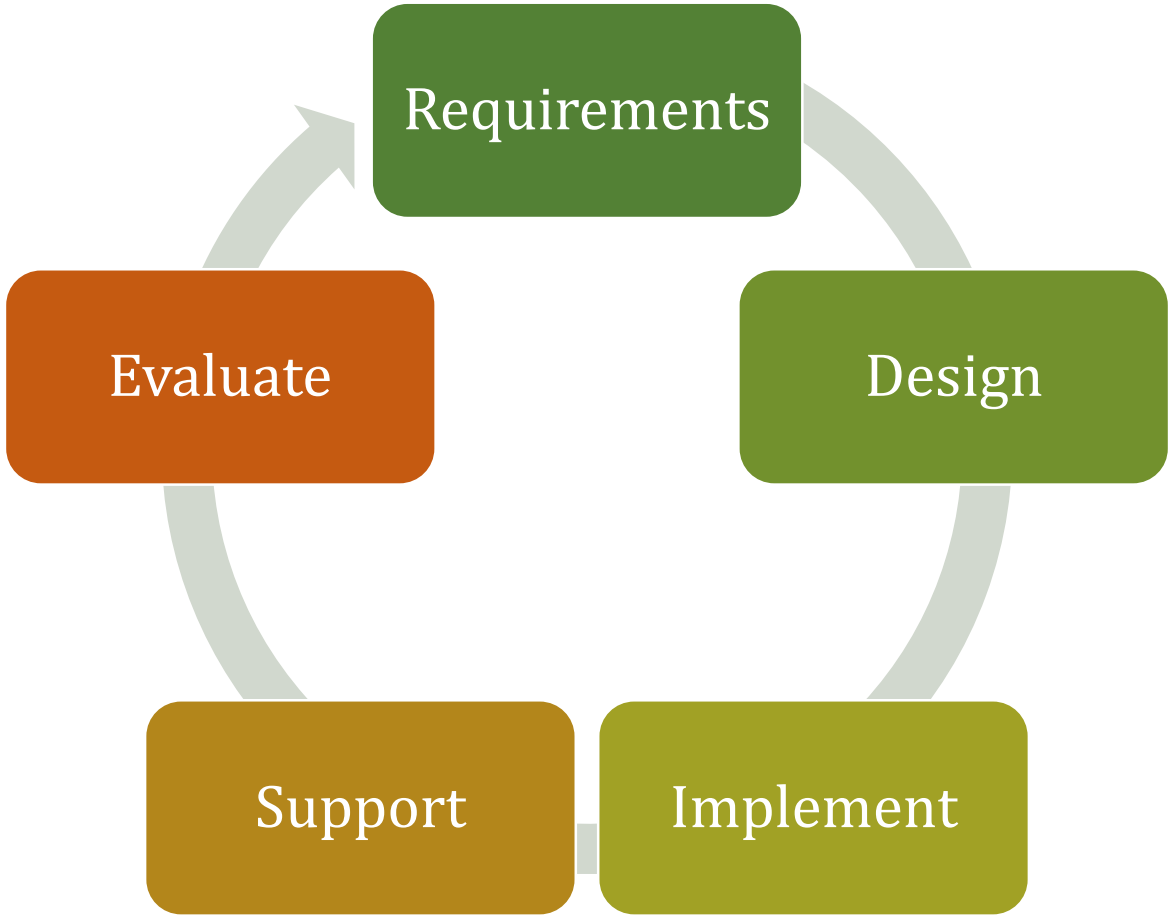
Software Design

Typical Life Cycle of an Application



Design starts from Specifications

- No good formal method of capturing specifications
- Figuring out HOW to meet the specifications
- Without violating the constraints from the specifications

The Software Design Problem

- Small projects can be managed in your head
  - You can design and remember all the details of the design without writing anything down
- Larger projects are too big to design in your head
  - Need a way to split up a big problem into smaller pieces

Design Concepts

- Abstraction – reducing information content, retaining only information which is most relevant
- Information hiding – Preventing implementation details from interacting with the abstract view
- Algorithm – Language independent method for solving a problem
- Modularity – Dividing a problem into smaller sub-problems
- Architecture – How modules interact with each other
- Stepwise Refinement – Modularizing the Modules
- Design Patterns – Previous solutions to similar problems

Example Design Process - Specifications

- Create a filter that replaces every occurrence of a “from” string with an occurrence of a “to” string
- “from” string represented as a parameter – no symbols or spaces
- “to” string optional parameter – no symbols or spaces
- Read input data from stdin
- Write output data to stdout
- Each input/output line less than 1024 bytes long

Design Step 1 - Abstraction

```
read parameters
while(read next line from stdin) {
        apply filter to single line
        write single line to stdout
}
return success
```

Design Step 2 – Design Pattern

How do I filter a single line (string)?

Alternatives:

- Treat the string as an array of characters
- Use standard library string functions

Design Step 3 – Stepwise Refinement

- Assuming array of characters…
- How do I check to see if the “from” string occurs in the “input” string?

input																					
M	a	r	y		h	a	d		a		l	I	t	t	l	e		l	a	m	b	\n	\0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

from				
M	a	r	y	\0
0	1	2	3	4

Design Step 4 - Algorithm

Q: Does “from” appear starting at position “i” in input?

```
assume match = true
for(k - position in from) {
        if (input[i+k] != from[k])
                match=false , break
}
if match is true, “from” appears at position i
```

Design Step 5 - Algorithm

```
for (i – position in “input”) {
        if (match(from,input[i]) {
                write “to” string to output
                move “i” past “from” string in input
        } else {
                copy input[i] to output
        }
}
```

Algorithm Using string functions

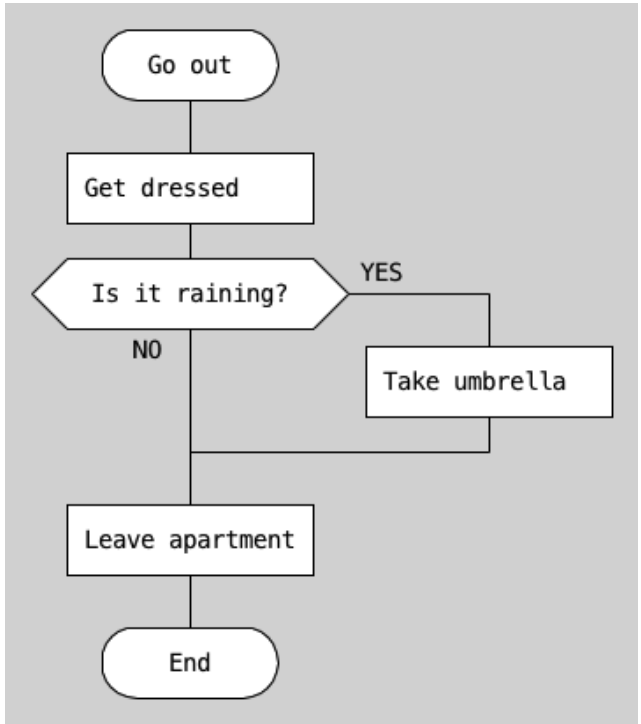
iptr – pointer to input string left (starts at input[0])

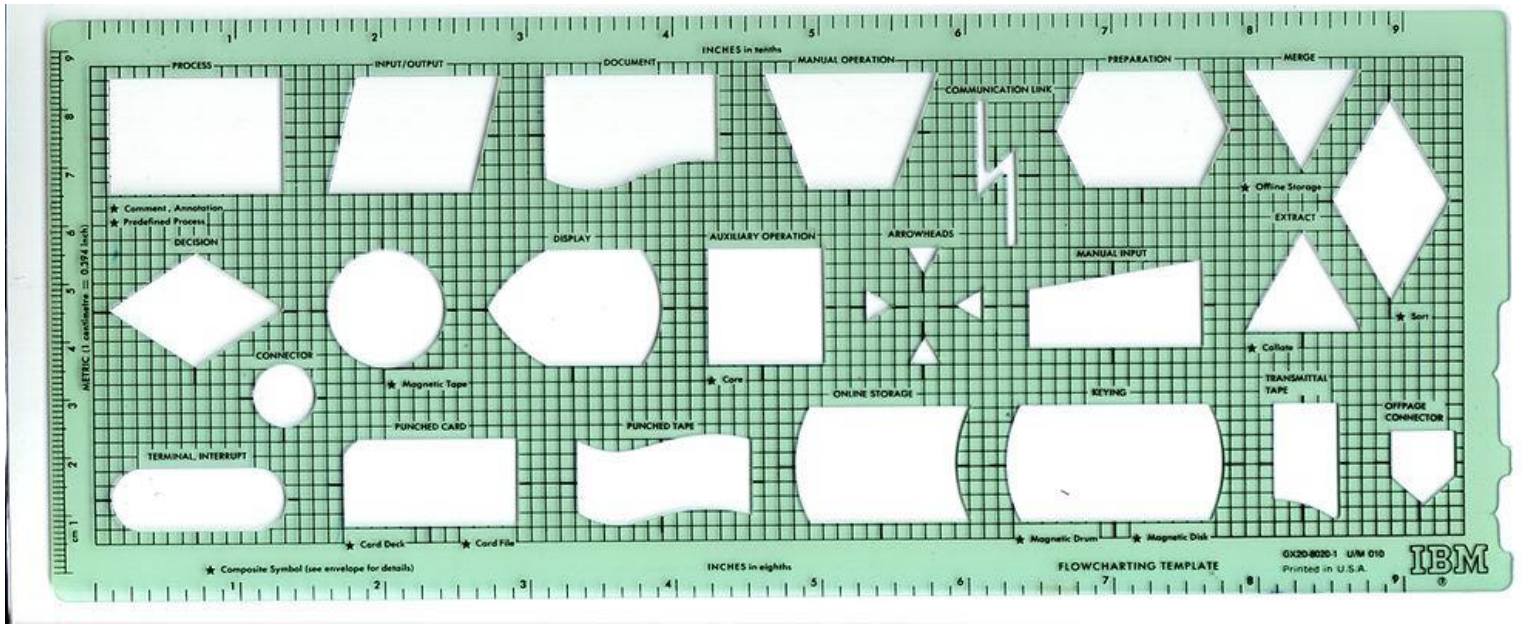
fptr – pointer to next match with “from” starting at iptr

```
while(fptr=strstr(from,iptr)) {
        write from iptr to fptr to output
        write “to” string to output
        increment iptr to fptr + “from”
}
write from iptr to end of input string to output
```

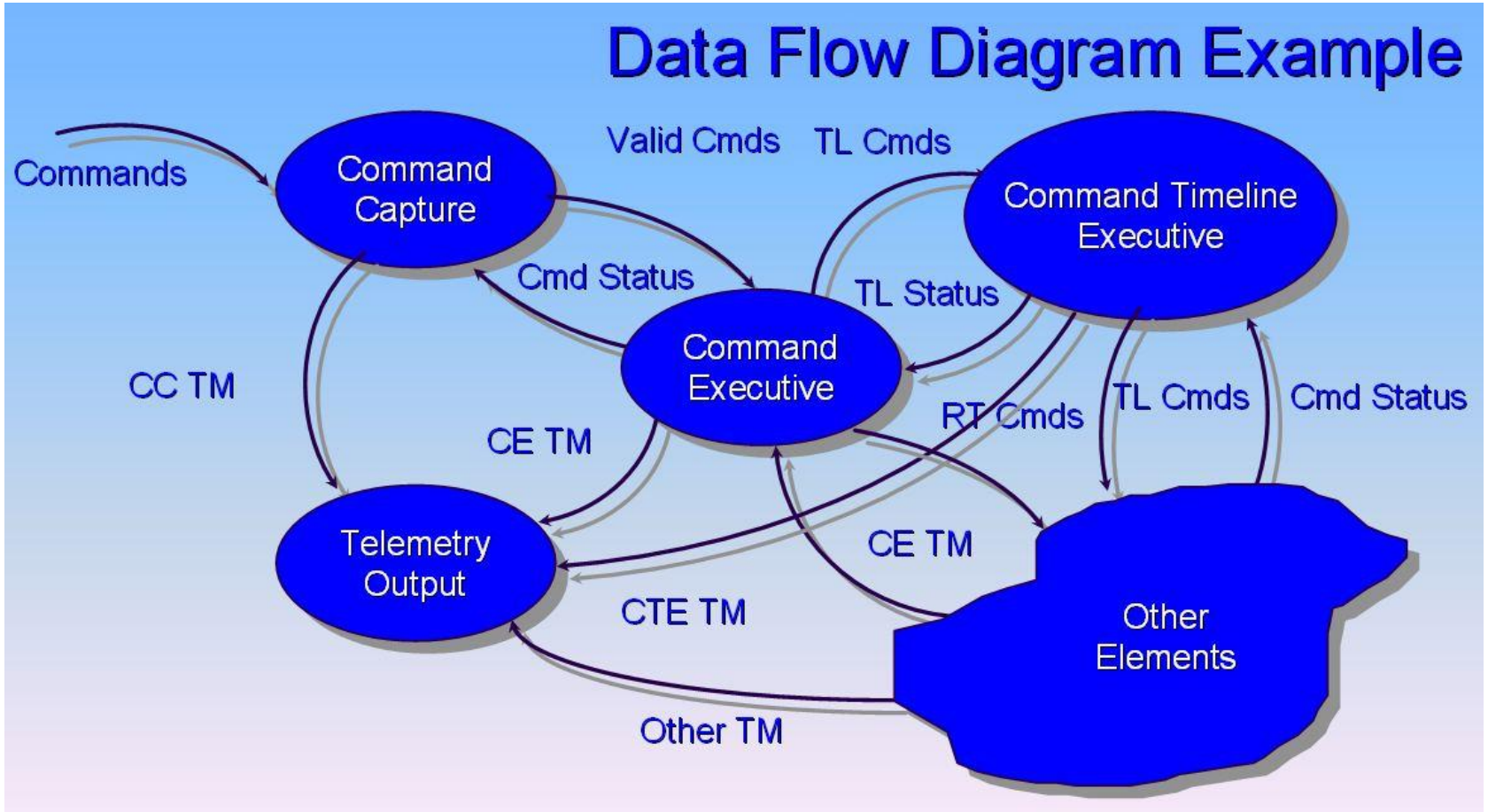
Documenting the Design

- Pseudo-Code
- FlowCharts





Designing the Data Flow



Data Structures

- We've seen several so far
  - Vectors, Matrices, Linked Lists, Trees
- Data Structure choice is part of the design
  - Need to know what data structures are good for
  - For instance, do I want to keep my input string as linked list?

Object Orientation

- Design Pattern – way of thinking about the world
- More next lecture.

Design Considerations

- Correctness
- Maintainability, Modularity, Usability, Performance
- Fault-Tolerance, Reliability, Robustness, Security
- Compatibility, Extensibility, Reusability, Scalability, Portability

Design Principles

- Consider Alternatives
- Cross-Link with the Specifications
- Make use of existing solutions
- Connect Design to the Real World
- Uniformity
- Accomodate change

Design Evolution

- Design Evolves over time
- Specifications Evolve
- Discovery during Design, Implementation, and Testing
- The later a design changes, the more it costs to fix!

- Extra Effort in design will save you time and effort!

Resources

- Programming in C, No references.
- Wikipedia Software Design https://en.wikipedia.org/wiki/Software_design
- Wikipedia Flowchart https://en.wikipedia.org/wiki/Flowchart
- Software Design Tutorials
  - http://www.tutorialspoint.com/software_engineering/software_design_basics.htm
  - http://www.yacoset.com/Home/how-to-design-a-computer-program