Slides

1. Table of Contents
2. Definitions
3. Large System Development Costs
4. Development Goals
5. Poor Communication
6. Design Communication
7. Design Elements
8. Structure Chart
9. Parameter Notation
10. Parameter Example
11. Flow Control: Selection
12. Flow Control: Selection (cont.)
13. Flow Control: Loop
14. Misc. Routine Calls
15. Connector Symbol
16. Interface Diagram
17. Global Data
18. Global X-Reference Charts
Definitions

Software Systems
- "An integrated whole composed of diverse, interacting, specialized structures and subfunctions." [IEEE]

Software Engineering
- Disciplined systematic technological activity for producing and maintaining software products by means of a controlled efficient process.

![Cost of Errors Diagram]

- Relative Cost
- Development Phases: Design, Code, Test
- Cost of Errors graph showing increasing cost with development phases.
### Cost vs. Number of Modules

![Graph showing the relationship between system development cost and the number of modules.](image)

The graph illustrates the following relationships:

- **Total Cost**: The overall cost of system development.
- **Cost per Component**: The cost associated with each individual module.
- **Interfacing Integration Cost**: The cost related to integrating components.

#### System Development Cost Relationships

<table>
<thead>
<tr>
<th>Project</th>
<th>Cost (millions)</th>
<th>Instructions (millions)</th>
<th>Effort (work years)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apollo Skylab</td>
<td>$209</td>
<td>23</td>
<td>6,000</td>
</tr>
<tr>
<td>NASA Satellite Range Monitoring</td>
<td>$30</td>
<td>1.25</td>
<td>1,000</td>
</tr>
<tr>
<td>FAA Air Traffic Control</td>
<td>$103</td>
<td>1.48</td>
<td>5,000</td>
</tr>
<tr>
<td>Safeguard ABM</td>
<td>$120</td>
<td>1.87</td>
<td>3,500</td>
</tr>
</tbody>
</table>
Development Goals

Programming
- Goal: Write Code

Software Design
- Goals
  Select components
  State Functions
  State Interfaces

Problem ▶ Code ▶ Solution

Problem ▶ Solution
Poor Communication

Developer

Code

??

Maintainer / Enhancer
Design Communication

1. Intro SE

- Idea (vague)
- Design Document
  - Concrete and Specific
  - Can be analyzed
  - Can be compared to implementation
  - Traceable through life cycle
A Design should contain
- Components
- Procedures
- Functions

Function of Each Component
- Suggestive names

Interfaces
- Control
  Calling Hierarchy
- Data
  Parameters
  Global Variables
  Files
Structure Chart

- A graphic tool used to hierarchically model the design solution of a problem.

Contains:
- Individual modules
- Data passed to/from modules
- Control Interfaces between modules

Does NOT contain a complete representation of the internal structure of individual modules.

Basic Elements

```
<table>
<thead>
<tr>
<th>Calling Module</th>
<th>GetWord</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Called Module</td>
<td>GetNext Char</td>
</tr>
</tbody>
</table>
```

Control Relationship
### Parameter Notation

**Parameter Direction Flow**

- **3 Types**
  1. **Input:** Value Parameters & Const Reference Parameters
  2. **Output:** Reference Parameters (Function changes parameter values independently of parameter’s original [passed] value.)
  3. **I/O:** Reference Parameters (Function changes parameter values dependent upon of parameter’s original [passed] value.)

**Parameter Classes**

- **2 Classes**
  1. **Data Parameter:** Any parameter upon which NO decision in the called module’s or calling module’s code is based.
  2. **Control Parameter:** Any parameter upon which a decision in the called module’s or calling module’s code is based.

---

**Diagram:**

```
GetWord

<table>
<thead>
<tr>
<th>Input Data</th>
<th>Output Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Parm</td>
<td>Input/Output Data</td>
</tr>
</tbody>
</table>

GetNext Char
```
1. Intro SE

Parameter Example

```c
void GetNextChar ( . . . ) {
    . . .
    GetChar (ch, uplow, terminator);
    if (terminator) . . .
}
```

```
void GetChar(char& chact,
            uplowtype uplowCase,
            bool& terminal )
```

**Note:** Function return values are treated as output parameters (list variable to which they are assigned).

Calls to multiple functions are usually listed left-right in order of execution.
void Move(...) {
    ...
    if (ch == plus) {
        DrawBox(row, col);
    }
    ...
}

Label decision with name(s) of variables used in decision.

What does the function header for DrawBox look like?
Select statements are diagrammed in a similar manner with multiple calls emanating from the decision diamond.
Flow Control: Loop

```c
void Drawbox (... ) {
  ...
  while (ch != plus)
  {
    ...
    DrawLine ( dir );
    ...
  }
  ...
}
```

Label with name(s) of variables used in decision.

Conditional Loop

Diagram:
- Draw Box
  - ch
- Draw Line
  - dir
Misc. Routine Calls

Recursive Routines:
- Routines that call themselves

Operating System Calls:

Predefined Module:
- (library routines)
Large designs span many pages
Interface Diagram

Interface Diagram #3

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Dir</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>data</td>
<td>in</td>
<td>Cartesian coords</td>
</tr>
<tr>
<td>y</td>
<td>data</td>
<td>in</td>
<td>of point</td>
</tr>
<tr>
<td>z</td>
<td>data</td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>scale</td>
<td>data</td>
<td>in</td>
<td>scale of axis</td>
</tr>
<tr>
<td>angrad</td>
<td>cont</td>
<td>in</td>
<td>angle degrees (rad)</td>
</tr>
<tr>
<td>angle</td>
<td>data</td>
<td>out</td>
<td>polar coords</td>
</tr>
<tr>
<td>dist</td>
<td>data</td>
<td>out</td>
<td>of point</td>
</tr>
</tbody>
</table>
Global Data (extern)

- Must be highly justifiable (semi-global)
  - separately compiled, module scope
- Operation
  - Assign
  - Update
  - Reference

Diagram:

- Copy
  - Get Word
  - Up Case
  - Print Word

- Assign
- word
- Reference
- UpDate
Global Variables introduce complexities that must be mapped and traceable.

## Global Structures

<table>
<thead>
<tr>
<th>Procedures</th>
<th>Form</th>
<th>Setting</th>
<th>Date</th>
<th>U</th>
<th>A</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetNext</td>
<td>R</td>
<td>U</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Advance</td>
<td>U</td>
<td>A</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PutNext</td>
<td>A</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset</td>
<td></td>
<td>A</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

U : Update    A : Assign    R : Reference