Table of Contents

- Definitions
- Large System Development Costs
- Development Goals
- Poor Communication
- Design Communication
- Design Elements
- Structure Chart
- Parameter Notation
- Parameter Example
- Flow Control: Selection
- Flow Control: Loop
- Misc. Routine Calls
- Misc. Routine Calls (continued)
- Connector Symbol
- Interface Diagram
- Global Data
- 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

- **Relative Cost**
- **Cost of Errors**

<table>
<thead>
<tr>
<th>Phase</th>
<th>1</th>
<th>10</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Development Phases
Cost vs. Number of Modules

System Development Cost Relationships

Cost vs. Number of Modules

<table>
<thead>
<tr>
<th>Project Name</th>
<th>Cost (Millions)</th>
<th>Instructions (Millions)</th>
<th>Effort (man years)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apollo Skylab</td>
<td>$209</td>
<td>23</td>
<td>6000</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 Safeguard ABM</td>
<td>$103</td>
<td>1.48</td>
<td>5,000</td>
</tr>
<tr>
<td></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
Poor Communication

Developer

Maintainer / Enhancer

Code

??
Design Communication

- Design Document
  - Concrete and Specific
  - Can be analyzed
  - Can be compared to implementation
  - Traceable through life cycle
Design Elements

- 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

Basic Elements

![Structure Chart Diagram]

- Calling Module
- Called Module
- GetWord
- GetNext Char
- 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
     / \    \
   /   \   /   \    
  Input Data  Output Data
    \   /    \   /    
     \ /     \ /     
       Control Parm  Input/Output Data
          \       /    
            \     /     
             GetNext Char
```

GetWord

Input Data  Output Data

Control Parm  Input/Output Data

GetNext Char
**Parameter Example**

```c
void GetNextChar
...
    GetChar (ch, class, terminator);
    if (terminator) ...

Get Next Char

void GetChar
(char& chact ,
 uplowtype clas,
 bool& terminal )
```
What does the function header for `DrawBox` look like?

```c
void Move
...
    if ( ch == plus )
        DrawBox(row, col);
...
```

Diagram:

- **Move**
  - Conditional Call
  - row
  - col
- **DrawBox**

```c
void DrawBox(row, col);
```
void Drawbox

... 
while (ch != plus) 
{
...
DrawLine ( dir );
...
}
...
Misc. Routine Calls

- **Recursive Routines**
  - Routines that call themselves
  
  ![Factorial](image)

- **Operating System Calls**

  ![Calendar](image)
  
  ![Date](image)
■ Parallel Processing
  - Methods
    ■ FORK / JOIN
    ■ PARBEGIN / PAREND

```
PARBEGIN
  ProcessA (...);
  ProcessB (...);
PAREND
```

■ Predefined Module (library routines)

```
Get File
  fexists
```
Large designs span many pages
### 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

- Global Data (extern)
  - Must be highly justifiable
  - Operation
    - Assign
    - Update
    - Reference
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>
</tr>
</thead>
<tbody>
<tr>
<td>GetNext</td>
<td>R</td>
<td>U</td>
<td></td>
</tr>
<tr>
<td>Advance</td>
<td>U</td>
<td>A</td>
<td>R</td>
</tr>
<tr>
<td>PutNext</td>
<td>A</td>
<td>R</td>
<td></td>
</tr>
<tr>
<td>Reset</td>
<td></td>
<td>A</td>
<td>A</td>
</tr>
</tbody>
</table>

U: Update  A: Assign  R: Reference