Chapter 10
Behind the Scenes: Building Applications
Information Systems

• System
  – A collection of pieces working together to achieve a common goal

• An information system includes
  – Data—electronic, paper forms, graphics
  – People
  – Procedures
  – Hardware/software

• System development life cycle (SDLC)
  – An organized process (or set of steps) used to develop systems in an orderly fashion
System Development Life Cycle

1. Problem/Opportunity Identification
2. Analysis
3. Design
4. Development & Documentation
5. Testing & Installation
6. Maintenance & Evaluation
Problem/Opportunity Identification

• All steps involve IS and user persons
• The existing system is evaluated.
  – Problems are defined.
  – New proposals are reviewed.
  – Decisions are made to proceed with the projects.
  – The process is documented.
  – Relevant problems/opportunities are defined.
Analysis

• A program specification (goals and objectives of the project) is developed.
• A feasibility assessment is performed.
• User requirements are defined.
• Analysts recommend a plan of action.
• Consultations required to answer new questions that arise.
• User signs off for approval of plan.
Design

- A detailed plan for programmers is developed.
- Flowcharts or pseudo-code and data-flow diagrams are used for the current and proposed logic of the system. More questions arise.
Development and Documentation

• Actual programming takes place that follows the flowchart or pseudo-code logic.
• More questions arise.
• First phase of the program development life cycle (PDLC).
• Development is documented.
• User documentation is created.
Testing and Installation

- Program is tested for proper operation and fixed as necessary.
- Program is installed for use.
- Testing and results are documented.
Maintenance and Evaluation

• Performance of the system is monitored.
• Corrections and modifications to the program are made.
• Maintenance procedures and results are documented.
• User training manuals produced.
Joint Application Development (JAD)

- Helps designers adapt to changes in program specifications
- Schedules are adjusted
- Includes customer involvement
- No communication delays
- Also referred to as:
  - Accelerated Design
  - Facilitated Team Technique
The Life Cycle of a Program

• Programming is the process of translating a task into a series of commands a computer will use to perform that task.

• Programming involves
  – Identifying the parts of a task the computer can perform
  – Describing tasks in a specific and complete manner
  – Translating the tasks into a language understood by the computer’s CPU
Program Development Life Cycle

Step 1
Describing the Problem
(The Problem Statement)

Step 2
Making a Plan
(Algorithm Development)

Step 3
Coding
(Speaking the Language of the Computer)

Step 4
Debugging
(Getting Rid of Errors)

Step 5
Finishing the Project
(Testing and Documentation)
Step 1: Describing the Problem

• The problem statement is:
  – The starting point of programming
  – A description of tasks the program is to accomplish
  – A description of how the program will execute the tasks
  – Created through interaction between the programmer and the user

• The program statement includes error handling, a testing plan, and output values.
# Parking Garage Example

**PROGRAM GOAL:** To compute the total pay for a fixed number of hours worked at a parking garage.

**INPUTS:**
- Number of Hours Worked .................................. a positive number

**OUTPUTS:**
- Total Pay Earned ........................................ a positive number

**PROCESS:**
The Total Pay Earned is computed as $7.32 per hour for the first eight hours worked each day. Any hours worked beyond the first eight are billed at $11.73 per hour.

**ERROR HANDLING:**
The input Number of Hours Worked must be a positive real number. If it is a negative number or other nonacceptable character, the program will force the user to reenter the information.

**TESTING PLAN:**

<table>
<thead>
<tr>
<th>INPUT</th>
<th>OUTPUT</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>8*7.32</td>
<td>Testing positive input</td>
</tr>
<tr>
<td>3</td>
<td>3*7.32</td>
<td>Testing positive input</td>
</tr>
<tr>
<td>12</td>
<td>8<em>7.32 + 4</em>11.73</td>
<td>Testing overtime input</td>
</tr>
<tr>
<td>-6</td>
<td>Error message/ask user to reenter value</td>
<td>Handling error</td>
</tr>
</tbody>
</table>
Step 2: Developing an Algorithm

• Algorithm development
  – A set of specific, sequential steps that describe what the program must do
  – Complex algorithms include decision points
    • Binary (yes/no)
    • Loop (repeating actions)
  – Visual tools used to track algorithm and decision points
Flowchart and Pseudocode

**Flowchart**

1. Ask for the number of hours worked.
2. Read the number of hours worked.
3. Check if the number of hours worked is less than or equal to 8.
4. If yes, compute the total pay without overtime. Otherwise, compute the total pay with overtime.
5. Print the total pay.
6. End.

**Pseudocode**

**Bold** terms show actions that are common in programming, such as reading data, making decisions, printing, and so on.

1. **Ask the user** how many hours they worked today.
2. If the number of hours worked is less than or equal to 8, compute **total pay** without overtime; otherwise, compute **total pay** with overtime.
3. **Print** **total pay**.

Underlined words are information items that appear repeatedly in the algorithm.
Top-Down Design

- Problem is divided into a series of high-level tasks
- Detailed subtasks are created from high-level tasks

```
GET INPUT
  Announce Program
  Give Users Instructions
  Read the Input
  NumberHoursWorkedToday

PROCESS DATA
  Determine If They Qualify For Overtime
  Compute Pay

OUTPUT RESULTS
  Print TotalPay

If (NumberHoursWorkedToday <= 8)
  Pay = $7.32 * NumberHoursWorkedToday
Else
  Pay = 7.32 * 8 + $11.73 * (NumberHoursWorkedToday - 8)
```
Object-Oriented Analysis

• Classes (categories of inputs) are identified.
• Classes are defined by information (data) and actions (methods or behaviors).
• Reusability is key.
Step 3: Coding

- Coding is translating an algorithm into a programming language
- Generations of programming languages

<table>
<thead>
<tr>
<th>Level</th>
<th>Language</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>1GL Machine</td>
<td>Bits: 1110 0101 1011 1111 0000 1011 1110 0110</td>
</tr>
<tr>
<td></td>
<td>2GL Assembly</td>
<td>Words: ADD Register 3, Register 4, Register 5</td>
</tr>
<tr>
<td>High</td>
<td>3GL FORTRAN, BASIC, C, Java</td>
<td>Symbols: TotalPay = Pay + OvertimePay; SELECT isbn, title, price, price*0.06 AS sales_tax FROM books WHERE price&gt;100.00 ORDER BY title;</td>
</tr>
<tr>
<td></td>
<td>4GL SQL</td>
<td></td>
</tr>
<tr>
<td>Natural</td>
<td>5GL PROLOG</td>
<td>Programmers can build applications <strong>without specifying an algorithm</strong>. Find all the people who are Mike’s cousins as: ?-cousin(Mike, family)</td>
</tr>
</tbody>
</table>
Compilation

- Compilation is the process of converting code into machine language. Converting program called a Compiler.
- The compiler reads the source code and translates it into machine language.
- After compilation, programmers have an executable program, usually saved on the hard drive.
The interpreter translates source code into a line-by-line intermediate form.

Each line is executed before the next line is interpreted.

Programmers do not have to wait for the entire program to be recompiled each time they make a change.

Programmers can immediately see the results of changes as they are making them in the code.

Requires less memory than compiling.

Allows for platform independence.
Coding Tools: Integrated Development Environments

• Editor: Special tool that helps programmers as they enter the code

• Debugging: Removal of errors in code
  – Syntax error: Mistake in use of the language
  – Logic error (runtime error): Mistake in the algorithm—you get wrong results
Step 4: Debugging

- Running a program to find errors is known as debugging.
- Sample inputs are used to determine runtime (logic) errors.
- Debugger: Tool that helps programmers locate runtime errors.
Step 5: Finishing the Project

• Users test the program (internal testing)
• Beta version released
  – Information collected about errors before final revision
• Software updates (service packs)
  – Problems found after commercial release
• Documentation created
  – User manuals
  – User training sessions
Programming Languages

• Selecting the right language
  – Space available
  – Speed required
  – Organizational resources available
  – Type of target application
Windows Applications: Visual Basic 2008

- Used to build Windows applications
- Object-oriented language
- Visual Basic 2008 is the current version

Visual Basic
C and C++

- **C**
  - Developed for system programmers at Bell Labs
  - Combines high- and low-level programming features
  - Modern operating systems are written in C

- **C++**
  - Uses the same features as C
  - Includes object-oriented design

**Sample C**

```c
#include <stdio.h>

int main()
{
    printf("This is output from my first program!\n");
    return 0;
}
```

**Sample C++**

```cpp
#include <iostream.h>

int main()
{
    cout<<"HEY, you, I'm alive! Oh, and Hello World!"
    return 0;
}
```
Java

- Created for Internet programming at Sun Micro
- Object-oriented features
- Large set of existing classes
- Architecture neutral
- Java applets: Small Java-based programs
Web Applications

• HTML/XHTML
  – Hypertext Markup Language/Extensible
  – Extensible means you can add commands
  – Hypertext Markup Language
  – Not a true programming language
  – Uses special symbols (tags) to control how Web pages are viewed

• Extensible Markup Language (XML)
  – Enables computers to efficiently transfer information between Web sites
Web Applications

• Scripting languages: Languages limited to performing a specific set of specialized tasks
  – JavaScript
    • Used to make Web pages more visually appealing and interactive
  – VBScript
    • Subset of VB used to add interactivity to Web pages
  – PHP
    • Another scripting language gaining in popularity—dynamic, graphics

• Dynamic decision making
  – Web page can display content based on user choices
Web Applications

• Active Server Pages (ASP) and Java Server Pages (JSP)
  – Add interactivity capabilities to Web pages
  – Translate user information into a request for more information from a company’s computer

• Flash
  – Enables elaborate animations to be created for Web pages

• XML—extensible markup language
  – Enables designers to define their own data-based tags
Adobe Flash and Microsoft SilverLight

• Flash
  – Used to develop Web-based multimedia
  – Includes its own scripting language, ActionScript

• SilverLight
  – Supports development of multimedia and interactive Web applications
The Next Great Language

• Large projects may take 30 minutes to compile
• Interpreted languages may become more important
  – Python--interactive
  – Ruby--interactive
  – Smalltalk
Blender *(Freeware)*

- Video game development tool
- Open source
- Built-in game engine
- Built-in physics engine
- Uses logic bricks to simplify programming