MIS 5203
Systems & Infrastructure Lifecycle Management 1

Week 7
Oct 13, 2014
Study Objectives

• Systems Acquisition
• Software Development
Systems Acquisition

- When to Buy?
  
  A. COTS (commercial off the shelf) system meets the need for most part, for example ERP system for HR or Finance
  
  B. COTS system requires limited customization
  
  C. COTS systems require limited integration with the enterprise system
  
  D. Don’t have in-house skill to build
  
  E. It’s an strategic system requiring rapid changes to gain strategic advantage
  
  F. Everyone should have 60% of the systems as the package (industry average)
  
  G. It’s a CRM (customer relationship management) system
Software Buy vs. Build Consideration (week 1)

- The timing of when the systems need to be functional
- The cost of development against that for buying
- Resource Skills and availabilities
- Licensure and Maintenance needs
- Compatibility with the Strategic Business Plans
- Compatibilities with Organizations IT Infrastructure
- Future Requirements
- Ability to Make changes
RFP (Week 1)

What’s the difference between RFI and RFP?

RFI (Request for Information) Stage
RD (Requirement Definition) Stage
RFP (Request for Proposal Stage)

Typical RFP Steps

1. Provide RFP to targeted Vendors
   - RD
   - Architecture
   - Expected time-line

2. Provide clarification
   - Provide clarification to the vendors on their questions

3. Receive Vendor Proposals
4. Vendor short-list and presentation
   - Presentation further refines responses

5. Vendor Selection
   Criteria include:
   - Time
   - Cost
   - Prior Experience and track record
   - Ability to make changes
Acquisition Steps of a Software Package

1. Review the needs and requirements
2. Acquire the package (RFP)
3. Customize the package to meet the company’s needs
4. Develop needed Interfaces to the enterprise systems
5. UAT
6. Deployment
RFP Evaluation

Factors influencing RFP Evaluation

• Functionality to be supported (analysis of the bid against requirements)
• Cost (initial, license, maintenance)
• Timeline of implementation (Turnaround time)
• Ease of upgrading the package (both software and hardware)
• Scalability of the systems against current and future demands
• Expected Response Time to correct outage or defects in production
• Vendor’s financial and stability
• Market share of the systems against competitive packages
• Customers testimonials
• Compatibility against existing systems
• Training and Hiring requirements
• Impact on systems and Network performance
• Technology platform (modern mainstream vs. obsolete)
• Security of the system and data
Contracts

• Legal Document
• Typically driven by SLA (Service Level Agreement)
  – Response Time
  – Error Rate
  – Error Resolution Timeline
  – Etc.
• Penalties and Rewards
  – When SLAs are not-met vs. exceeded
Outsourcing

- Is outsourcing same as the systems acquisition?

- Similar to the Acquisition of a software package in RFP and evaluation

- It’s different as in this case enterprise software development is done by an outsources vendor
Outsourcing Model

• “Managed Service” type engagement provides more control to the Vendor on the development, design, and management decisions
• “Staff Augmentation” type engagement provides more control to the enterprise on the development, design, and management decisions

Note: Any of the above two models can involve an offshore partner (Global Outsourcing) to
• Reduce the cost
• Take advantage of the “sun” (working round the clock)
Software Development

- What do Developers do?
  - A. Program Design
  - B. Code
  - C. Unit Test
  - D. Debug
  - E. Code profiling
  - F. Resolve Testing Issues
  - G. Fix defects found in production
  - H. Deploy software in production
  - I. Write Business requirements
  - J. Write systems requirements
# Programming Languages

<table>
<thead>
<tr>
<th>Type</th>
<th>Detail</th>
</tr>
</thead>
</table>
| 1<sup>st</sup> Generation Languages (1GL) | • Machine Language (early 1950’s)  
• Written as hardware instructions  
• Machine could interpret but not easily comprehensible by people                                                                                     |
| 2<sup>nd</sup> Generation Languages (2GL) | • Assembly Language (late 1950’s)  
• Tedious but much improved compared to 1GL                                                                                                                                                           |
| 3<sup>rd</sup> Generation Languages (3GL) | • Assembly Language (started in 1960’s) and still being used  
• Introduced English-like statements like “if-then” “go to”  
• Object-Oriented concepts introduced in 1980  
• Fortran, Cobol, C, Visual Basic  
• Small Talk, C++, Java, Visual Basic, C# (Object Oriented)  
• Has evolved over time, with respect to ease of use, memory allocation, existing libraries  
• C# and Java are mainstream languages today                                                                                                    |
# Programming Languages

<table>
<thead>
<tr>
<th>Type</th>
<th>Detail</th>
</tr>
</thead>
</table>
| 4\textsuperscript{th} Generation Languages (4GL) | • Database Management system (DBMS) provides SQL (structured Query Language). For example “SELECT From Table1, Field A B C, where A="myname”  
  • Template or model based coding (create a model), which generates code/pseudo code  
  • FOCUS, dBase, Natural |
| 5\textsuperscript{th} Generation Languages (5GL) | • Artificial intelligence system  
  • “Fuzzy logic” or “neural weighting algorithms” provides decisions based on likelihood |

Each generation of language as provided significant improvements over the previous stage with respect to human effort.

3GL languages are considered high-level general purpose programming languages, and most commonly used language today. 3GL languages are “compiled” (compilers translated the language instruction to lower level machine instructions), so than machine can understand them.
Programming Language

- HTML (HyperText Mark-Up Language) is an scripting language commonly used by browsers such as Internet Explorer
- Other scripting languages examples - PERL, Javascript, Tcl, Python, PHP. Scripting languages are interpreted and do not require compilation
- Functional Languages examples – LISP, PROLOG. They work faster in certain scenario where logical comparison is needed such as (“OR” and “AND” logic)
**Evolution of Programming Language (PL)**

- PL evolution continues
- Today’s Internet Programming use dynamically typed languages for speed of getting the programs written
- Static typed languages are more structured, less error prone, but require compilation hence slow
- New languages are more flexible, e.g., C# 4.0 added dynamically typed feature, Dart (JavaScript replacement by Google, Swift (Objective C replacement by Apple)

The continuum of development languages.
Adapted from “Gradual Evolution, Dynamically Typed Languages Adopt Features of Static Typing to Cope up with growth,” by Neil Savage, Communication of the ACM, Oct 2014
Programing Language (C) Example

```c
#include "stdafx.h"
#include "stdio.h"

int _tmain(int argc, _TCHAR* argv[])
{
    int yourMarks=0, waiting=0;

    printf("Hello MIS5103 Students\n");
    printf("Hi there. We will program a simple Math\n");
    printf("Please enter your quiz marks: ");
    scanf("%d", &yourMarks);

    if (yourMarks > 88)
    {
        yourMarks = yourMarks +5; /* you got 5 additional marks for class participation */
        printf("You got an A.Your final marks is: ");
        printf("%d", yourMarks, "\n");
    }

    else if (yourMarks > 75)
    {
        yourMarks = yourMarks +5; /* you got 5 additional marks for class participation */
        printf("You got a B.Your final marks is: ");
        printf("%d", yourMarks, "\n");
    }

    else
    {
        printf("Let us talk how you can do better \n");
    }

    scanf("%d", &waiting);
    return 0;
}
```
3GL Programming Language

How Does the Program Work?

1. Write the Code
   - One or more files

2. Compile the Code
   - Compiled code

3. Link the code
   - Linked code with other libraries

4. Create Executable
   - This is the code we deploy

5. Run the Code
   - Executable code can run
Debugging Example Using Visual Studio IDE /Class Demo

![Visual Studio IDE with debugging session](image)
Debugging

• What’s Debugging?

• Allows “bugs” or program defects to be easily found and removed

• Concept of “break point” “watch point” “step in” “step over”

• Can see the flow of control during “run time”

• Can see memory dumps, which provides picture of actual memory at run time
IDE

• What’s an IDE (Integrated Development Environment)?

• IDE allows writing code, compiling, building, debugging, etc. using one tool

• Examples: Visual Studio, Eclipse, NetBean

• Can support multiple languages

• Can pre-generate Skelton code for many templates

• Adds to productivity by showing programming errors and providing prompts while writing code

• Creates both debug and deployable code and solution
Program Design

• Designing of the Program so that they are structured well
• Multiple routines, procedures, classes need to be written. Many times a procedure calls another to do its work
• Would like to avoid poor structure, where multiple programs call each other, which makes understanding of the control and process flow difficult
• Better written program follow the principle of “more cohesion” and “less coupling”
Logging

- Allows different level of transaction and program execution logs
- The level of the logging could be one of the following with increased level of logging
  - Fatal
  - Error
  - Informational
  - Debug
- Logging can be turned on and off in production by configuration changes, which helps in troubleshooting
Unit Testing

• Developers use unit testing to test the code that they build
• This testing focus on the each code component that is being written or changed
• Unit testing may not cover all the functional scenarios
Code Profiling

• Code profiling is used to check the efficiencies of the code
• Memory leak or performance of the code can be found by profiling
• Commercial tools such as Jetbrain, Fortify, Checkstyle
Auditing Development

- Ensure transaction traceability and audit trail of the atomic transactions
- Verify integrity of key calculations and processes
- Verify that the system can process erroneous data correctly (by providing the correct error messages, or by identifying and then correcting the incorrect data while processing)
- Verify unit-test was done to ensure the written code works as expected
- Verify that “code profiling” was done to improve quality of the code such as identifying and correcting memory leaks, performance issues etc.
- Ensure the traceability of the code against and the detailed design and systems requirements for completeness
- Verify if there are “best practices” included as part of the project such as peer-review of the code
Question

What advantages of the integrated development environment (IDE) would have compared to standalone development?

A. It prevents errors during Design (SDLC phase 3)
B. It eliminates most of the work needed during the requirement phase (SDLC phase 2)
C. It makes programming simple and helps debug program code
D. It eliminates the testing needs (SDLC phase 4)
Upcoming Assignments/Tests

1. Individual Case Study -2 (Requirements and Use Case): Mon 10/20 before the class
2. Quiz 2 (Week 5 – Week 8): Mon 10/27
3. Group Case Study -2 (Requirements): Mon 11/3 before the class
Summary of Today’s Class

• Systems Acquisition
• Evaluation of RFP
• Development of Code
• Programming Languages
• Unit Testing
• IDE, Debugging, Programming Techniques, Logging, Profiling
• Focus of the Next Class and Reading
• Questions