1. Introduction to Information Systems and Development Management

What is Information System?

A set of components that work together to collect, process, store, and share information.

6 components of Information Systems

- Hardware
- Software
- Data Sources
- Telecommunications
- Process
- Human Expertise

Thought to ponder: AI still can't replace most human experts in IT (at least for now).

Why? Humans possess a unique ability to empathize, interpret emotions, and cater to the user experience. Humans produce user-centric solutions for human users that not even AI or robots can come close to replicating. *(Ayha nata mangurog siguro sa future. But for now, all is still well.)*

Information Systems Analysis and Design

- The process that teams follow to create or improve an information system.

Application Software

- The product of information systems analysis and design.

Who is a System Analyst?

- An IT expert who bridges the business side and technical side.

System Development Methodology

- A standard process followed in an organization to conduct all the necessary steps to analyze, design, implement, and maintain information systems.

Systems Development Life Cycle (SDLC)

A common methodology for systems development in many organizations, features several phases that mark the progress of the systems analysis and design effort.

5 Phases of SDLC:

1. Planning
2. Analysis
3. Design
4. Implementation
5. Maintenance

Phases and its Deliverables

Planning

- Setting priorities for systems and projects
- System Architectures
- Selection of technologies and tools
- Detailed steps and workplans
- System scope, resources, business cases

Analysis

- Description of current system
- Analysis of problems and opportunities
- Recommendations of fixes, improvements, or replacements

Design

- Functional, detailed specifications of all system elements
- Technical, detailed specifications of all system elements
- Acquisition of plan for new technology

Implementation

- Code, documentation, training procedures, support capabilities

Maintenance

- New versions or releases of software with associated updates to documentation, training, and support

5 Software Process

1. Communication
2. Planning
3. Modeling
4. Construction
5. Deployment

Software Engineering Layers

- Tools
- Methods
- Process
- A quality focus

Software Engineering Practices

1. Understand the problem
2. Plan a solution
3. Carry out a plan
4. Examine the result for accuracy

Project Management

- The practice of planning, organizing, and overseeing the execution of a project from start to finish.
- Involves defining project goals, identifying tasks, and resources required to achieve them, setting timelines, and monitoring progress to ensure the project is completed on time, within budget, and to stakeholders' satisfaction.
- Involves various processes and methodologies, including risk assessment, stakeholder management, and communication management. It is essential for ensuring the success of projects in various fields, including software development, construction, marketing, and many others.

2. System Development Life Cycle and Software Process

Software Process

The set of related activities and associated outcome that produce a product

4 Fundamental Process Activities

1. Software specification
2. Software development
3. Software validation
4. Software evolution

Software Process Models

Generic, high-level, abstract definitions of software processes that can be used to explain different approaches to software development.

- Waterfall Model
- Agile Model
- Iterative Model

Factors to consider in choosing the right software process models

- Project size and complexity
- Stability of requirements
- Customer involvement and feedback
- Team size and expertise
- Schedule and time constraints

McCall's Software Quality Factors

Product Revision

- Maintainability
- Flexibility
- Testability

Product Transition

- Portability
- Reusability
- Interoperability

Product Operation

- Correctness
- Reliability
- Usability
- Integrity
- Efficiency

3. Agile Development

Plan-driven Software Development

- Completely specifies requirements
- Design, build, and test
- Identifies separate stages in the software process with outputs

- Iteration occurs within activities with formal documents used to communicate between stages of the process

Agile Development

- Produce software quickly
- Incremental but in small increments
- Consider design and implementation to be the central activities in the software process
- Iteration occurs across activities

Common Practices of Agile Development

- Customer Collaboration
- Iterative Development
- Flexibility
- Self-organizing Teams
- Continuous Improvement

Key Methodologies/Frameworks of Agile Development

Kanban

A visual approach where teams use a board to track tasks through different stages of development

Scrum

For small teams that involve sprints. Developed by Ken Schwaber and Jeff Sutherland in the 1990s.

Roles:

- Scrum Master
- Product Owner
- Development Team

Sprints and Artifacts:

- Sprints
- Product Backlog
- Sprint Backlog
- Sprint Burndown Chart

Extreme Programming (XP)

Stronger technical emphasis. Focused on improving team collaboration through five core values:

- Communication
- Simplicity
- Feedback
- Courage
- Respect

Key XP Practices:

- Test-Driven Development
- Code Refactoring
- Continuous Integration
- Pair Programming
- Coding Standards

Adaptive Project Framework

Designed for projects where unexpected changes can occur. Focuses on managing the resources currently available, rather than those initially planned.

Adaptive Software Development

Focused on continuous adaptation, involves three overlapping phases:

- Speculate
- Collaborate
- Learn

Dynamic Systems Development Method (DSDM)

Emphasizes full project cycle with a more structured approach. It consists of four main phases:

- Feasibility and Business Study
- Functional Model or Prototype Iteration
- Design and Build Iteration
- Implementation

Feature-Driven Development (FDD)

Combines agile best practices with a focus on specific software features. Relies heavily on customer input to prioritize features and for frequent updates.

12 Principles of Agile Methods

1. Satisfy Customers Early and Often
2. Embrace Changing Requirements
3. Deliver Value Frequently
4. Promote Collaboration

5. Build Around Motivated Individuals
6. Face-to-face Communication
7. Prioritize Working Software
8. Maintain a Sustainable Pace
9. Continuous Excellence
10. Keep It Simple
11. Foster Self-Organizing Teams
12. Reflect and Adjust

Issues of Scaling Agile Development:

Coordination Across Teams

- **Challenge:** Agile practices, like Scrum or XP, are often designed for small, co-located teams. Scaling these methods to large organizations with multiple teams requires effective coordination and communication.
- **Impact:** Misalignment between teams can lead to integration issues and inconsistencies.

Maintaining Agile Principles

- **Challenges:** As teams grow, maintaining the core Agile principles, such as flexibility and rapid iteration, becomes harder.
- **Impact:** Large-scale projects may become less agile and more rigid, deviating from Agile's adaptive nature.

Complexity in Communication

- **Challenges:** With more teams involved, communication becomes more complex, making it difficult to ensure that all teams are aligned and informed.
- **Impact:** Increased risk of miscommunication and delays in decision making.

Consistency in Practices

- **Challenges:** Implementing scaling frameworks like SAFe (Scaled Agile Framework), LeSS (Large Scale Scrum), or Spotify model can be complex and requires careful adaptation to fit organizational needs.
- **Impact:** Poor implementation can lead to confusion and resistance from teams.

Issues with Combining Agile with Plan-Driven Approaches

Conflicting Cultures

- **Challenges:** Agile focuses on flexibility and iterative development, while plan-driven approaches emphasize detailed upfront planning and documentation.
- **Impact:** Conflicting priorities and methodologies can lead to friction between teams and management.

Integration of Processes

- **Challenges:** Combining Agile's iterative cycles with plan-driven approaches' structured phases can result in mismatched processes and workflows.
- **Impact:** Difficulty in integrating practices may lead to inefficiencies and delays.

Documentation Vs. Flexibility

- **Challenges:** Agile minimizes documentation, while plan-driven methods require comprehensive documentation.
- **Impact:** Balancing the need for documentation with Agile's focus on working software can be challenging.

Resource Allocation

- **Challenges:** Agile requires adaptive resource management, whereas plan-driven approaches often rely on fixed resource plans.
- **Impact:** Misalignment in resource allocation can cause delays and resource conflicts.

Change Management

- **Challenges:** Agile welcomes change, but plan-driven approaches may resist changes once the plan is set.
- **Impact:** Integrating these approaches can result in resistance to change and difficulty in adapting to new requirements.

4. Origins of Software

1. Early Days (1940s - 1950s)

- Computers were huge, expensive, and programmed using machine language (binary 1s and 0s)
- Later, assembly language made it slightly easier using short codes instead of pure binary
- Software was often bundled with hardware
- Electronic Numerical Integrator Computer
- Universal Automatic Computer

2. The Birth of High-Level Languages (1950s - 1960s)

- Formula Translation (FORTRAN)
    - a high-level language primarily used for scientific numerical computing
- Common Business-Oriented Language (COBOL)
    - a high-level language primarily used for business, finance, and administrative systems

3. Software Industry Emerges (1970s - 1980s)

- Personal Computer
- Operating System
- The First Spreadsheet Program

4. Modern Software Era (1990s - Today)

- GUI replaced text-only screens
- Rise of Internet-based software
- Mobile apps and AI-powered tools dominate today’s software landscape