Course Overview

- This module aims to provide students with sound understanding and experience in Development methodologies, techniques and tools. Agile and rapid approaches to developing information systems and software are critical for responding to changing business environment. The module will examine in detail different methodologies for rapid development and their application in practice. Specific industry strength type of technology for RAD will be utilised for solving real-world problems.
- Lecturer: Dr. Franklin Leung

  (email: leungfranklin33@gmail.com)
- composition of marks: Project-50%(Group Interim+ Pair Assignment + Group Final Presentation and Demonstration), 1 hour Phase Test (50%)

Overview of COM302

SDLC

Team Building

Agile Development Overview

Behaviour Driven Modelling

Test Driven Modelling

Version Control & Team Collaboration Tools

NodeJS & JSON

Visual Studio Agile Development Tool

Emerging technologies

Topic 1:
System Theory

Systems Theory

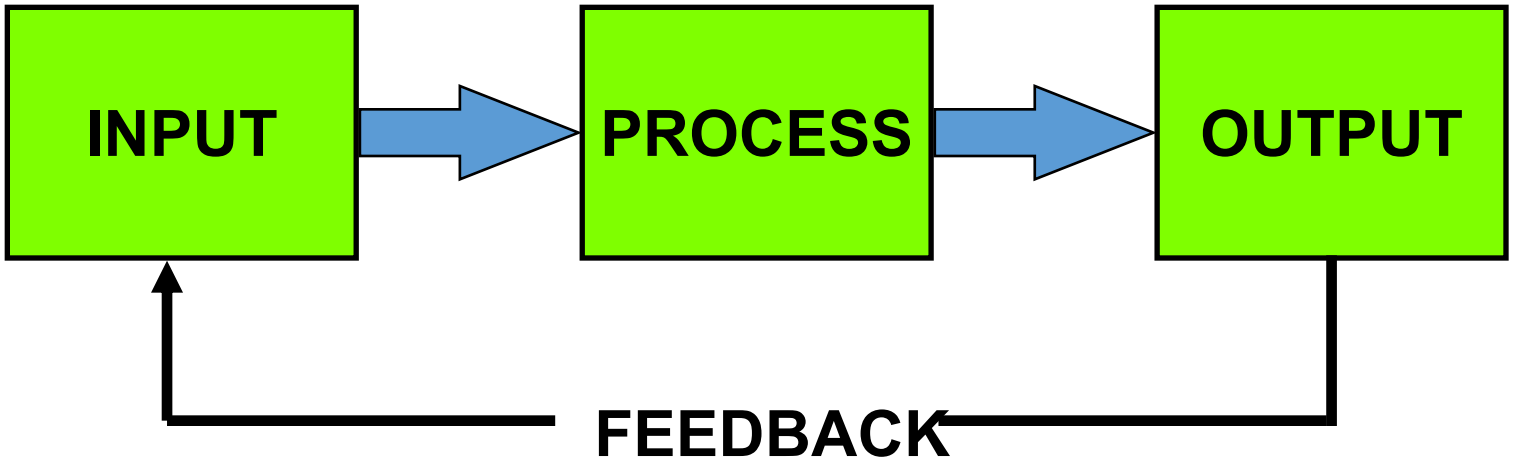
- What is a System?
- Systems and Subsystems
- Business organization as a Socio-technical system
- Information Systems (IS) as a system

What is a System?

- A collection of parts that work together to achieve a goal/task
  - Examples
    - Solar system
    - Digestive systems
    - Public transport system
    - Central heating system
    - Computer system
    - Information system

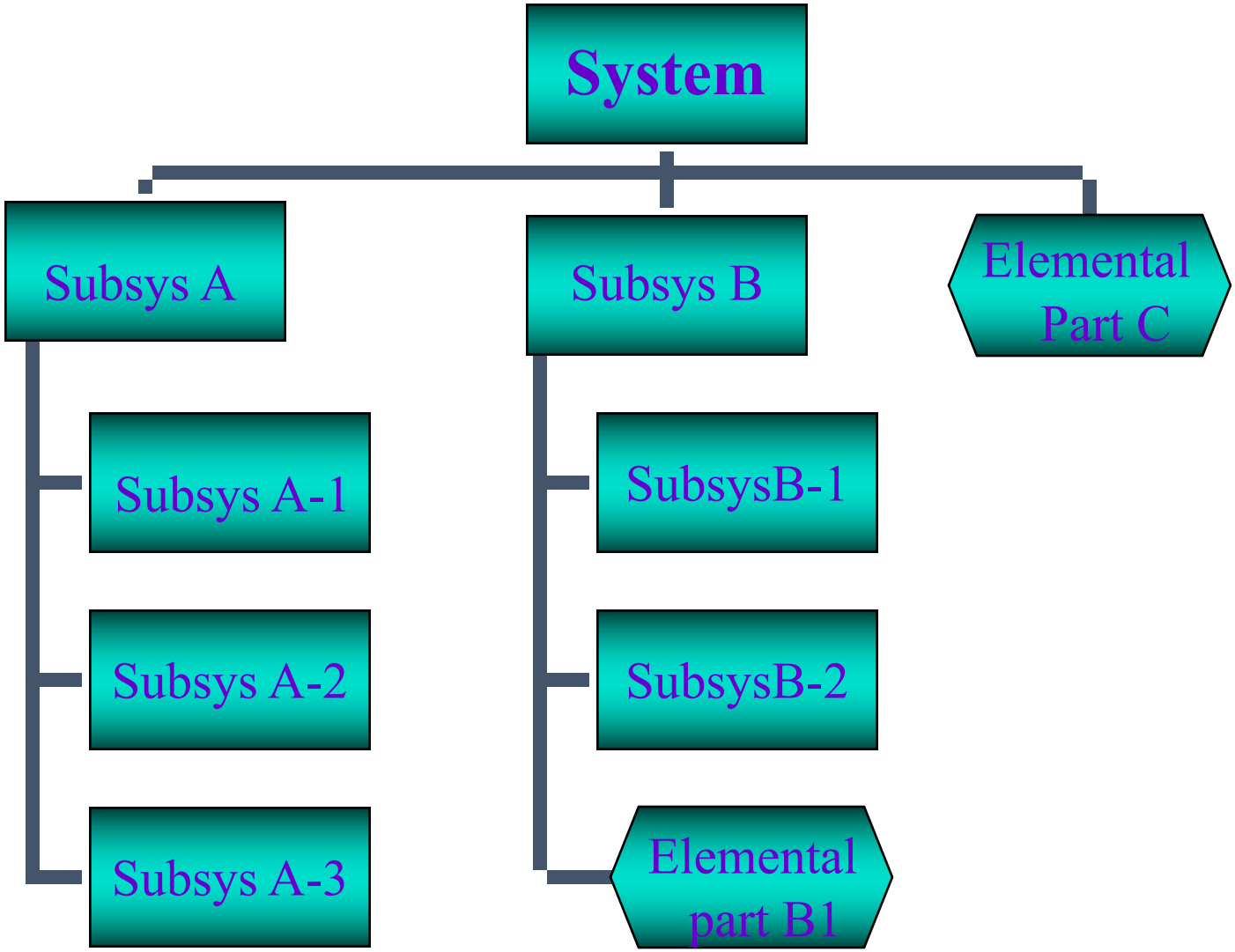
- A set of objects and relationships among the objects viewed as a whole and designed to achieve a purpose

System Elements



Systems

Can Be Composed of Subsystems



What is subsystem?

- A subsystem is simply a system within a system.
  - Automobile is a system composed of subsystems:
    - Engine system
    - Body system
    - Frame system
  - Each of these subsystem is composed of sub-sub -- systems.
    - Engine system: carburetor system, generator system, fuel system, and so son

Bad Systems

- Fail to meet requirements
- Poor performance
- Poor reliability
- Lack of usability
- Example difficulties:
  - Not to schedule
  - Not to budget
  - Runaway = 100% over budget or schedule
- Some problems are simply “wicked” problems

Reasons for Failure

- Complexity
  - Shifting requirements
  - Bad estimation
  - Bad management
  - New technology
- Must tackle complexity by, for example:
  - Structure partitioning of problem
  - Organized interaction of parts
  - Ensure you achieve the task
- Systems are subject to the need for continuing change

Important System Concepts

- Decomposition
  - The process of breaking down a system into smaller components
  - Allows the systems analyst to:
    - Break a system into small, manageable subsystems
    - Focus on one area at a time
    - Concentrate on component pertinent to one group of users
    - Build different components at independent times

Important System Concepts

- Modularity
  - Process of dividing a system into modules of a relatively uniform size
  - Modules simplify system design
- Coupling
  - Subsystems that are dependent upon each other are coupled
- Cohesion
  - Extent to which a subsystem performs a single function

System Analysis and Design (SAD)

- Systems Analysis: understanding and specifying in detail what an information system should do
- System Design: specifying in detail how the parts of an information system should be implemented
- Definition of SAD:
  - The complex organizational process whereby computer-based information systems are developed and maintained.

System Analysis and Design (SAD)

- Analysis: defining the problem
  - From requirements to specification
- Design: solving the problem
  - From specification to implementation

Why is it important?

- Success of information systems depends on good SAD
- Widely used in industry - proven techniques
- Part of career growth in IT - lots of interesting and well-paying jobs!
- Increasing demand for systems analysis skills

Views of Systems Analysis

- How to build information systems
- How to analysis information system needs
- How to design computer based information systems
- How to solve systems problems in organizations

Topic 2: System Development Methodology

System development methodology

- A standard process followed in an organization to conduct all the steps necessary to:
  - Analyze
  - Design
  - Implement
  - Maintain

  information system

Systems Development Life Cycle (SDLC)

- It is a common methodology for systems often follows for system development in many organization, featuring several phases that mark the progress of the systems analysis and design effort.

- SDLC phases:
  - 1-Project identification and selection
  - 2-Project initiation and planning
  - 3-Analysis
  - 4-Design
    - 4.1Logical design
    - 4.2Physical design
  - 5-Implementation
  - 6-Maintenance

1.19

1-Project identification and selection phase

- The first phase of the SDLC in which an organization total information systems needs are identified analyzed, prioritized and arranged.
  - Identifying Potential development projects
  - Classifying and ranking projects
  - Selecting projects for development

Cont.
1- Project identifying and selection

- This stage is critical to the success of the rest of the project.
- People:
  - Users, analyst, system managers coordinating the project
- Activities:
  - Interviewing user management, summarizing the knowledge obtained estimating the scope of the project and documenting the result
- Output:
  - Feasibility report: problem definition and summarizing the objectives

2-Project initiation and planning phase

- The second phase of the SDLC in which a potential IS project is explained and an argument for continuing with the project is presented. A detailed plan is also developed for conducting the remaining phases of the SDLC for the propose system. Output are:
  - Detailed step – work plan - high level system requirement – assignment of team members

3-Analysis phase

- The third phase of the SDLC in which the current system is studied and alternative replacement systems are proposed.
  - Description of current system
  - Where problem and opportunities are with a general recommendation on how to fix, enhance or replace current system

Cont.
3- Analyzing systems needs

- The primary objective of the analysis phase is to understand and document the business needs and the processing requirements of the new system. There are six primary activities in this phase:
- Gather information.
- Define system requirements.
- Build prototypes for discovery of requirements .
- Prioritize requirements.
- Generate and evaluate alternatives.
- Review recommendations with management

4-Design phase

- The forth phase of the SDLC in which the description of the recommended solution is converted into logical and then physical system specification.
  - Logical design:
    - The part of the design phase of the SDLC in which all functional feature of the system chosen for development in analysis are described independently of any computer platform.
  - Physical design:
    - The part of the design phase of the SDLC in which the logical specification of the system from logical design are transformed into technology specific details from which all programming and system construction can be accomplished.

4.1 Logical design output

- Functional,
- Detailed specification of all system elements
  - Input
  - Output
  - Process

4.2 Physical design output

- Technical
- Detailed specification of all system elements
  - programs,
  - files,
  - network,
  - system software
  - etc
- Acquisition plan of a new technology

Cont.
4- Designing the recommended system

- Its primary objective is to convert the description of the recommended alternative solution into system specification.
    - High-level design consists of developing an architectural structure for software programs, databases, the user interface, and the operating environment.
    - Low-level design entails developing the detailed algorithms and data structures that are required for program development.

- Seven major activities must be done during design:
    - Design and integrate the network.
    - Design the application architecture.
    - Design the user interfaces .
    - Design the system interfaces.
    - Design and integrate the databases.
    - Prototype for design details.
    - Design and integrate the system controls

5- Implementation

- The fifth phase of the SDLC in which the information system is
  - Coded,
  - Tested,
  - Installed, and
  - Supported in the organization.
- Outputs:
  - Code, documentation, training procedures and support capabilities

6-Maintenance

- The final phase of the SDLC in which the information system is systematically repaired and improved

- Output are:
  - New versions of releases of software with associated updates to documentation, training, and support

Topic 3: Improving the traditional SDLC

Disadvantages of traditional SDLC

- It is too expensive (cost + time) when dealing with change once it is developed
- It is structured approaches that requires to follow all its phases
- Maintains costs are too expensive

Improving the traditional SDLC

1. Structured analysis and structured design
2. Object oriented analysis and design
3. Prototyping
4. Joint Application Design (JAD)
5. Participatory design

1- Structured analysis and structured design

- More focus on reducing maintenances and time effort in system development
- Integrate change when needed

2- Object Oriented Analysis and Design (OOAD)

- A more recent approach to system development that is becoming is object oriented analysis and design (OOAD).
- It is often called third approach to system development, after the process oriented and data oriented approaches
- Definition: OOAD
  - It systems development methodologies and techniques base on objects rather than data or process

Object, Inheritance and object class

- Object:
  - A structure that encapsulates (packages) attributes and methods that operate on those attributes. An object is an abstraction of a real world thing in which data and processes are placed together to model the structure and behavior of the real world object
  - Combine data and processes (called methods) into single entities called Object

Object class

- Group of objects that have the same attributes and behavior
- A set of objects that share a common structure and a common behavior (methods)

Inheritance

- The property that occurs when entity types or object classes are arranged in a hierarchy and each entity type or object class assumes the attributes and methods of its ancestors.

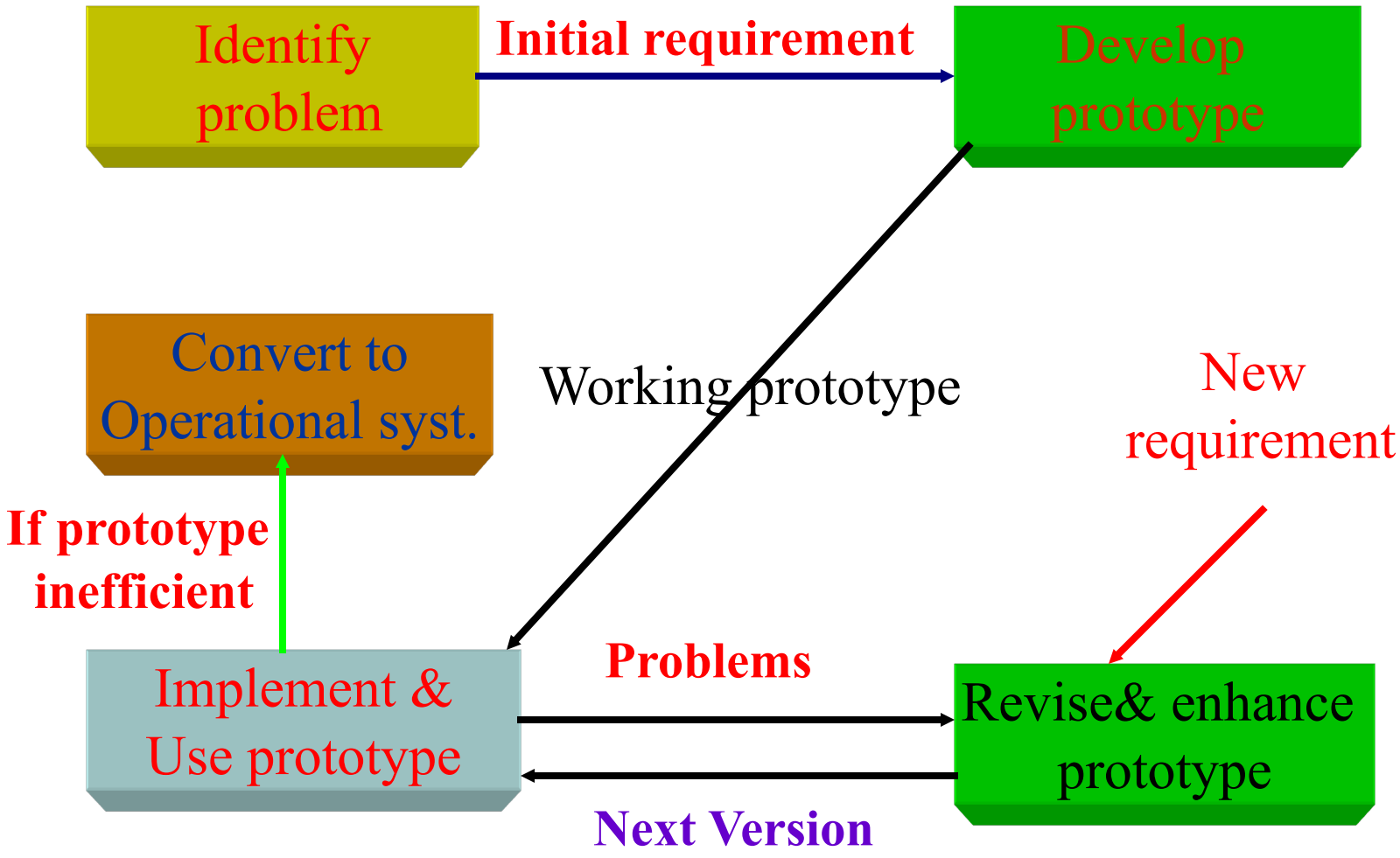
3- Prototyping

- An iterative process of systems development in which requirements are converted to a working system that is continually revised through close work between an analyst and users.
- You can build prototype by some development tool to simplify the process.
  - CASE: Computer Aided Software Tools such as Oracle (designer 2000)
  - 4GLs: fourth-generation languages
- Prototyping is a form of Rapid Application Development (RAD)

RAD disadvantages

1. RAD may overlook software engineering principles,
2. Resulting in inconsistencies among systems modules,
3. Noncompliance with standards, and
4. Lack of system component reusability

The prototype methodology



The prototype methodology

1. The analyst work with team to identify the initial requirement for the system:
2. The analyst then build the prototype. When a prototype is completed, the users work with it and then tell the analyst what they like and do not like about it.
3. The analyst uses this feedback to improve the prototype
4. Take the new version back the users
5. Repeat (2-4) until the users satisfied

Prototype advantages

1. Prototyping involves the user in analysis and design
2. its ability to capture requirements in concrete rather than abstract form
3. To being used stand alone
4. It is may be used to augment the SDLC

When I use Prototype

1. User requirements are not clear
2. One or few users and other stakeholders are involved with the system
3. Possible designs are complex and require concrete form to fully evaluate
4. Communication problem have existed in the past between user and analysts
5. Tools and data are readily available to rapidly build working systems

4-Joint Application Design (JAD)

- In the late 1970 systems development personnel at IBM developed a new process for collecting IS requirements and reviewing system design. It is called JAD
- Definition: It is structured process in which users, mangers, and analysts work together for several days in a series of intensive meeting to specify or review system requirements

5- Participatory design

- End users are involved in the SD around a table in one room to agree about system requirements and system design
- They responsible about the freeze of design "Milestone"

Automated Tools and Technology

- Analyst rely on automated tools to:
  - Increase productivity
  - Communicate more effectively with users
  - Integrate the work that they do on the system from beginning to the end of the life cycle
- Examples:
  - Computer-Aided Systems Engineering (CASE -tools)
  - Application Development Environments (ADE -tools)
  - Process and Project Managers

Improve productivity of IS development

- Computing technology can be used to improve productivity. CASE tools, for example, provide many productivity enhancing capabilities, such as code generation, diagramming tools, and screen and report designing tools.

Topic 5:
Software
- Its Nature and Qualities

Outline

- Software engineering (SE) is an intellectual activity and thus human-intensive
- Software is built to meet a certain functional goal and satisfy certain qualities
- Software processes also must meet certain qualities
- Software qualities are sometimes referred to as “ilities”

Software product

- Different from traditional types of products
  - intangible
    - difficult to describe and evaluate
  - malleable
  - human intensive
    - involves only trivial “manufacturing” process

Classification of sw qualities "ilities"

- Internal vs. external
  - External→ visible to users
  - Internal→ concern developers
- Product vs. process
  - Our goal is to develop software products
  - The process is how we do it
- Internal qualities affect external qualities
- Process quality affects product quality

Correctness

- Software is correct if it satisfies the functional requirements specifications
  - assuming that specification exists!
- If specifications are formal, since programs are formal objects, correctness can be defined formally
  - It can be proven as a theorem or disproved by counterexamples (testing)

The limits of correctness

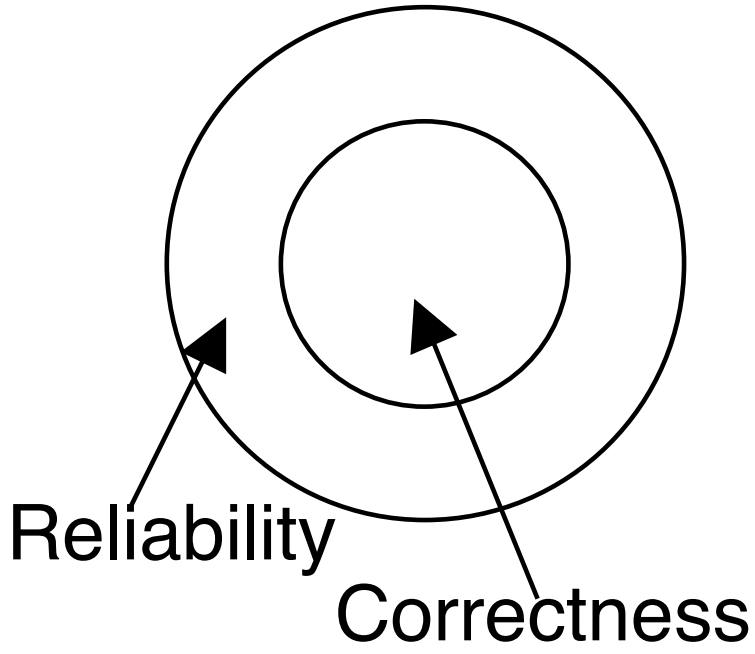
- It is an absolute (yes/no) quality
  - there is no concept of “degree of correctness”
  - there is no concept of severity of deviation
- What if specifications are wrong?
  - (e.g., they derive from incorrect requirements or errors in domain knowledge)

Reliability

- Reliability
  - informally, user can rely on it
  - can be defined mathematically as “probability of absence of failures for a certain time period”
  - if specs are correct, all correct software is reliable, but not vice-versa (in practice, however, specs can be incorrect …)

Idealized situation

- Requirements are correct



Robustness

- Robustness
  - software behaves “reasonably” even in unforeseen circumstances (e.g., incorrect input, hardware failure)

Performance

- Efficient use of resources
  - memory, processing time, communication
- Can be verified
    - complexity analysis
    - performance evaluation (on a model, via simulation)
- Performance can affect scalability
    - a solution that works on a small local network may not work on a large intranet

Usability

- Expected users find the system easy to use
- Other term: user-friendliness
- Rather subjective, difficult to evaluate
- Affected mostly by *user interface*
    - e.g., visual vs. textual

Verifiability

- How easy it is to verify properties
  - mostly an internal quality
  - can be external as well (e.g., security critical application)

Maintainability

- Maintainability: ease of maintenance
- Maintenance: changes after release
- Maintenance costs exceed 60% of total cost of software
- Three main categories of maintenance
  - *corrective*: removing residual errors (20%)
  - *adaptive*: adjusting to environment changes (20%)
  - *perfective*: quality improvements (>50%)

Maintainability

- Can be decomposed as
  - Repairability
    - ability to correct defects in reasonable time
  - Evolvability
    - ability to adapt sw to environment changes and to improve it in reasonable time

Reusability

- Existing product (or components) used (with minor modifications) to build another product
  - (Similar to evolvability)
- Also applies to process
- Reuse of standard parts measure of maturity of the field

Portability

- Software can run on different hw platforms or sw environments
- Remains relevant as new platforms and environments are introduced (e.g. digital assistants)
- Relevant when downloading software in a heterogeneous network environment

Understandability

- Ease of understanding software
- Program modification requires program understanding

Interoperability

- Ability of a system to coexist and cooperate with other systems
  - e.g., word processor and spreadsheet

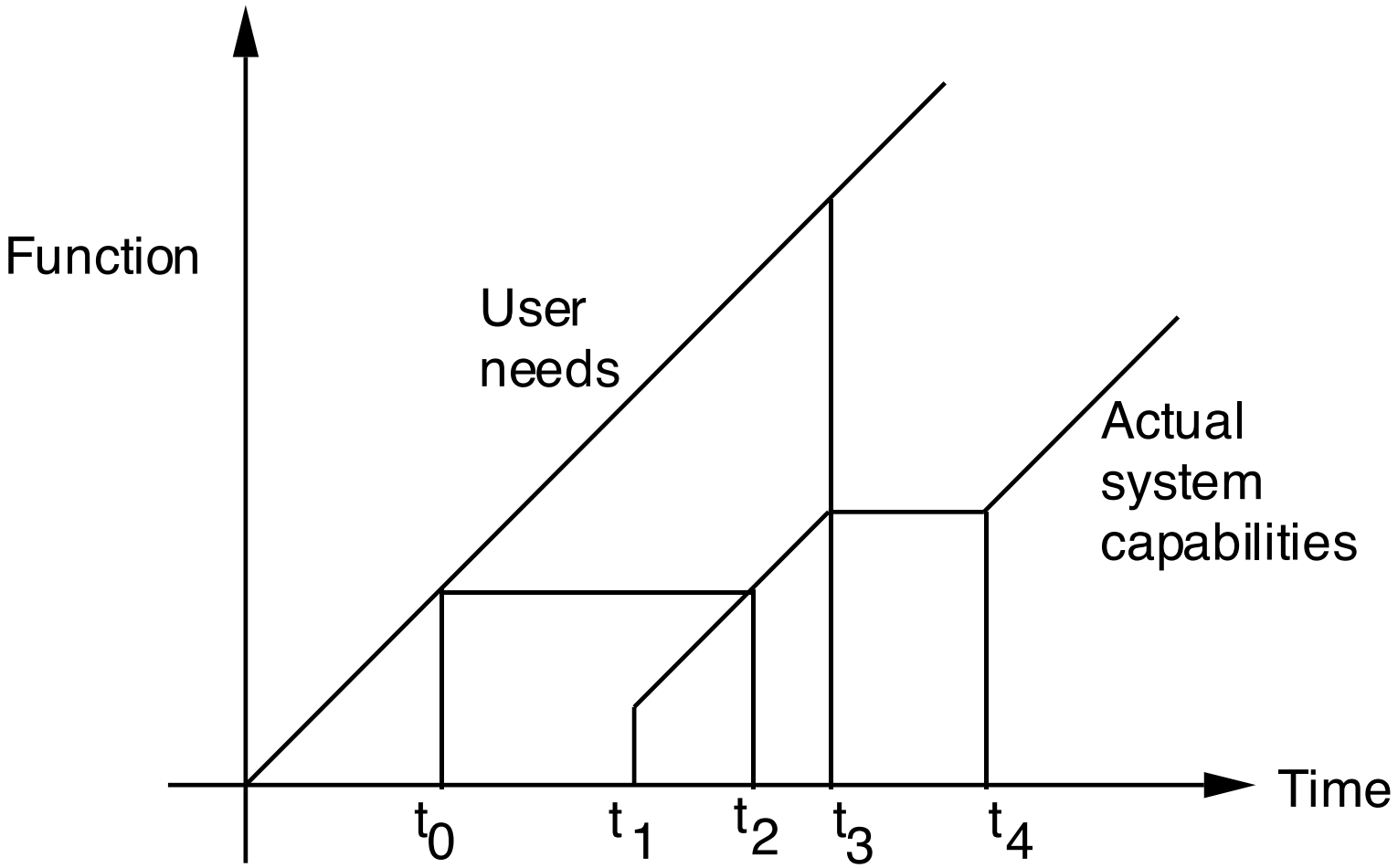
Typical process qualities

- Productivity
  - denotes its efficiency and performance
- Timeliness
  - ability to deliver a product on time
- Visibility
  - all of its steps and current status are documented clearly

Timeliness: issues

- Often the development process does not follow the evolution of user requirements
- A mismatch occurs between user requirements and status of the product

Timeliness: a visual description of the mismatch



Application-specific qualities

- E.g., information systems
  - Data integrity
  - Security
  - Data availability
  - Transaction performance.

Quality measurement

- Many qualities are subjective
- No standard metrics defined for most qualities

Topic 6:
IS Development and Team Building

Summary of approaches to system development

- There are three strategies of IS development
  1. Process-oriented approach
  2. Data-oriented approach
  3. Object-oriented approach

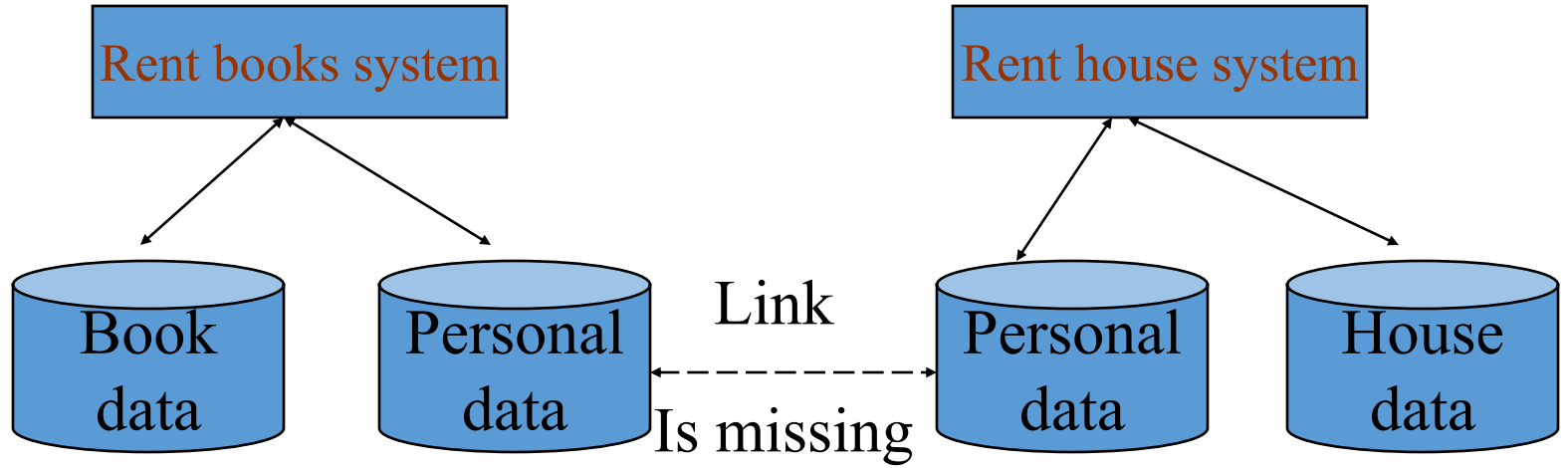
- Process-oriented approach
  - An strategy to IS development that focuses on how and when data are moved through and changed by an IS
- Data-oriented approach
  - An strategy to IS development that focuses on the ideal organization of data rather than where and how data are used.
- Object-oriented approach
  - A system development methodologies and techniques base on objects rather than data or process

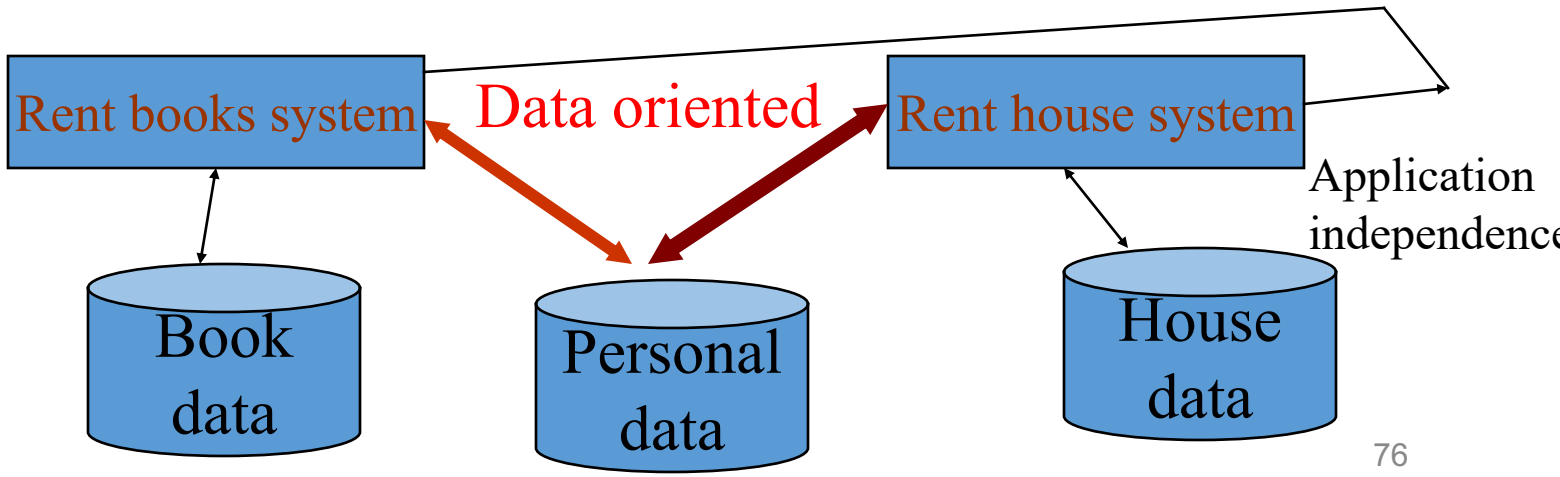
Application independence

- The separation of data and the definition of data from the applications that the use these data

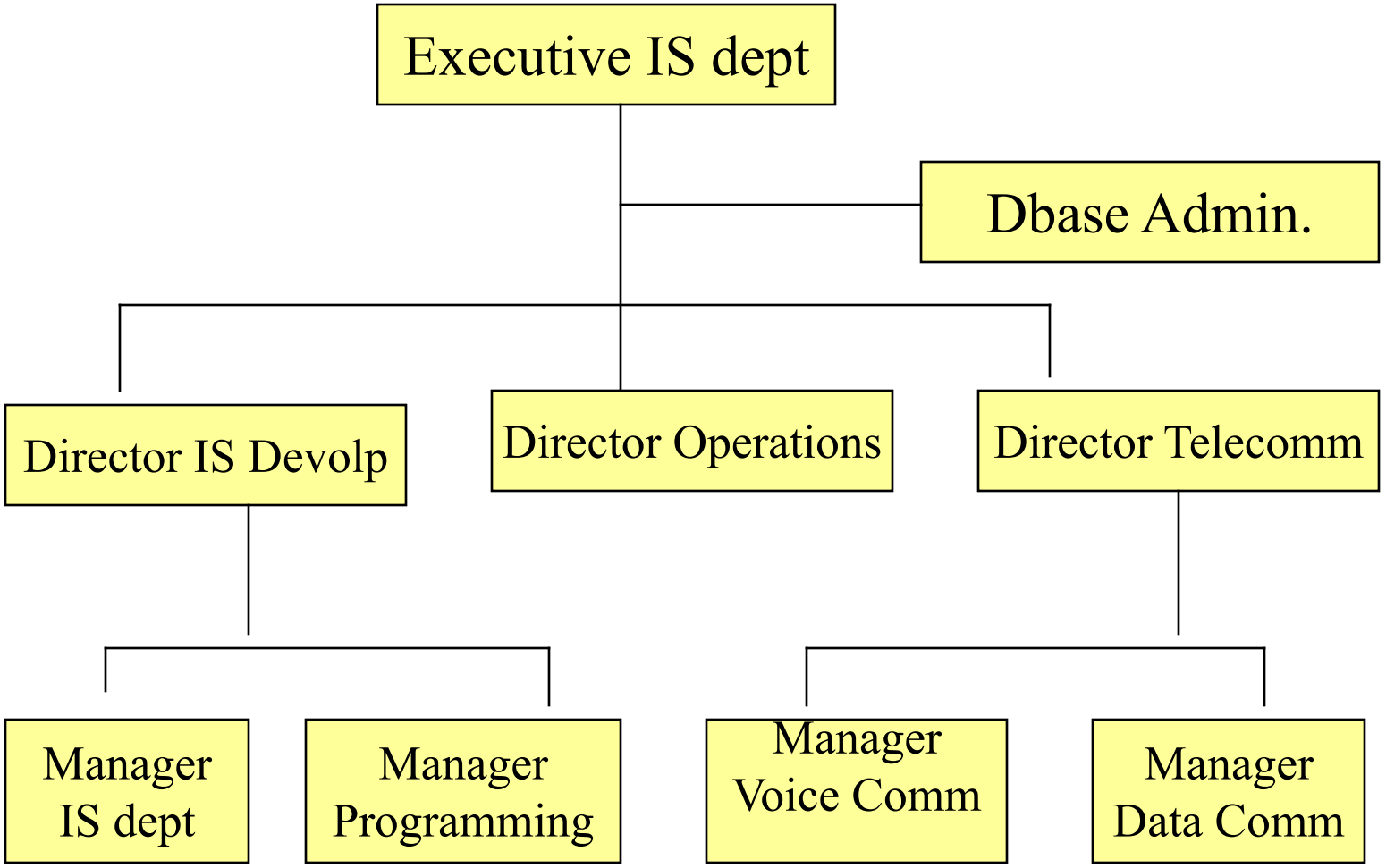
Example

Process Oriented Approach





Your Role and Other Organizational responsibilities in systems development



Stakeholders: Players in the Systems Game

- A stakeholder is any person who has an interest in an existing or new information system. Stakeholders can be technical or nontechnical workers.

Stakeholders Classification

- For information systems, the stakeholders can be classified as:
  - IS manager
  - Systems analysts in systems development
  - Programmers in systems development
  - End user in systems development
  - Supporting End user development
  - Business managers in systems development
  - Other IS managers/Technicians in system development

IS Manager in Systems Development

- The manager of an IS department may have a direct role in the systems development process if the organization is small or I that is the manger's style
- IS managers are more involved in allocating resources to and overseeing approved system development projects rather than in the actual project development process.
- There are several IS managers in any medium to large IS department.
  - The manager of an entire IS department may have the title Chief Information Officer and may report to the president or chairman of the firm.
  - Each division of the IS department will also have a manger
    - Director of IS development, IS operation manger, IS programmer director, etc.

Systems Analysts

- Systems analysts are the key individuals in the systems development process.
- A systems analyst studies the problems and needs of an organization to determine how people, data, processes, communications, and information technology can best accomplish improvements for the business.
- The organizational role most responsible for the analysis and design of information systems.

Skills of a Successful Systems Analyst

- Analytical skills
  - Understanding of organizations.
  - Problem solving skills
  - System thinking
    - Ability to see organizations and information systems as systems
- Technical skills
  - Understanding of potential and limitations of technology.

1.82

Skills of a successful systems analyst

- Managerial skills
  - Ability to manage projects, resources, risk and change
- Interpersonal skills
  - Effective written and oral communication skills
  - Help you work with end user as well as other system analysts and programmers

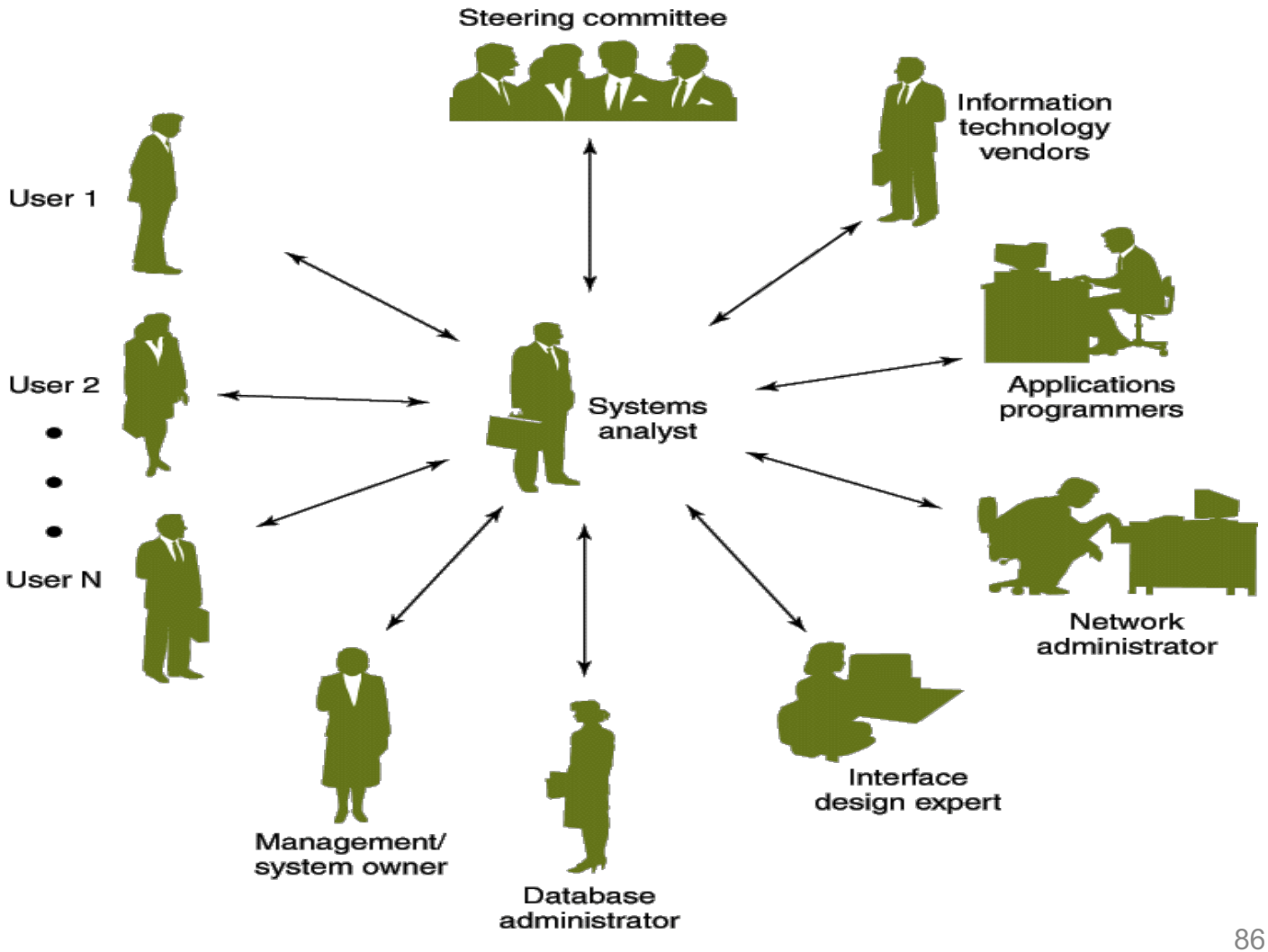
The analyst is responsible

- for:
  - The efficient capture of data from its business source,
  - The flow of that data to the computer,
  - The processing and storage of that data by the computer, and
  - The flow of useful and timely information back to the business and its people.

Variations on the Systems Analysts Title

- A business analyst is a systems analyst that specializes in business problem analysis and technology-independent requirements analysis.
- A programmer/analyst includes the responsibilities of both the computer programmer and the systems analyst.
- Others
  - Systems consultant
  - Systems engineer
  - Information engineer

The Systems Analyst as a Facilitator



Skills Required by Systems Analysts

- Working knowledge of information technology
- Computer programming experience and expertise
- General business knowledge
- Problem-solving skills
- Interpersonal communication skills
- Interpersonal relations skills
- Flexibility and adaptability
- Character and ethics
- Systems analysis and design skills

Programmers in systems development

- Programmers convert the specifications given to them by the analysts into instructions the computer can understand.
- Coding: writing a computer program
- Code generators have been developed to generate code from specifications, saving an organization time and money.
- The aim of CASE tools (Computer-Aided Software Engineering) is to provide a variety of code generators that can automatically produce 90% or more from the system specifications normally given a programmer.

Business managers in system development

- Another group to system development efforts is business managers such as functional department heads and corporate executives.
- These managers are important because they have the power to fund development projects and to allocate resources necessary for projects success.

Other IS managers/ Technicians in system development

- Database ----- database administrator
- Network and telecommunications experts:
  - Manager of Data Communication
  - Manager of Voice Communication
- Internal auditors

Characteristics of successful teams

- The characteristics are diversity in backgrounds,
  - skills, and goals;
  - tolerance of diversity, uncertainty, and ambiguity;
  - clear and complete communication;
  - trust;
  - mutual respect and putting one's own views second to the team;
  - A reward structure that promotes shared responsibility and accountability.

What is Teamwork & Team Building

Teamwork

- Concept of people working together as a team

Team player

- A team player is someone who is able to get along with their colleagues and work together in a cohesive group

Team Building

- Process of establishing and developing a greater sense of collaboration and trust between members

Why Should We Be a Team?

- When staff use their skills and knowledge together, the result is a stronger agency that can fulfill its mission

  "To provide accurate information that would assist individuals in achieving a better quality of life."

- People working together can sustain the enthusiasm and lend support needed to complete the work of each program.

How does a Team Work Best?

A Teams succeeds when its members have:

- **a commitment to common objectives**
- **defined roles and responsibilities**
- **effective decision systems, communication and work procedures**
- **good personal relationships**

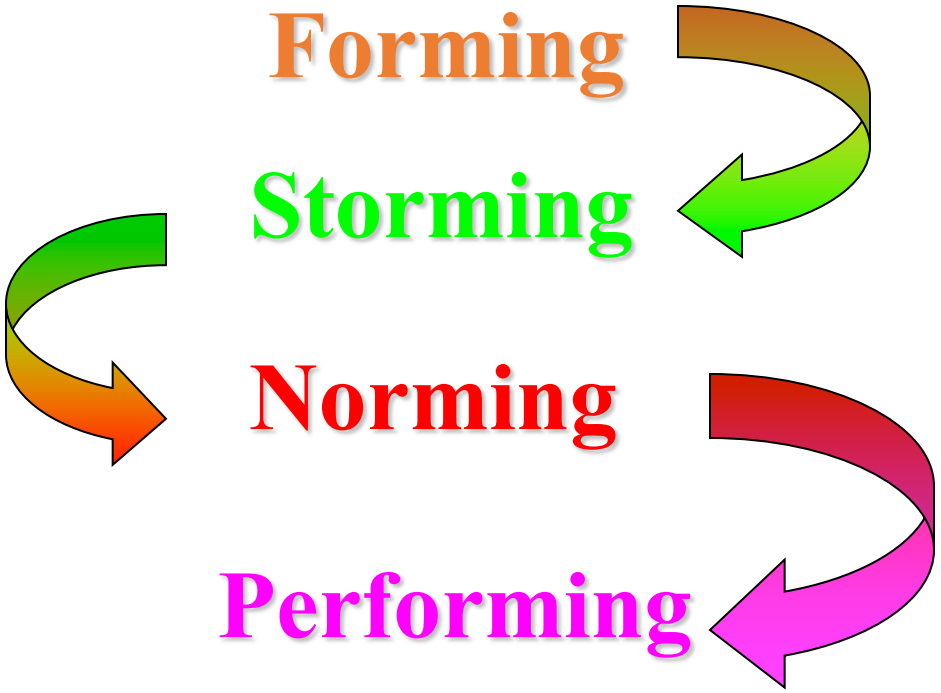
Team Morale Depends On

- Support
- Resources
- Communication
- Personalities

Teamwork Skills

- **Listen**
- **Question**
- **Persuade**
- **Respect**
- **Help**
- **Share**
- **Participate**

Stages in Team Building



Stage 1: FORMING

The Team

- defines the problem
- agrees on goals and formulates strategies for tackling the tasks
- determines the challenges and identifies information needed
- Individuals take on certain roles
- develops trust and communication

Team Roles - Leader

- **Encourages and maintains open communication**
- **Leads by setting a good example**
- **Motivates and inspires team members**
- **Helps the team focus on the task**
- **Facilitates problem solving and collaboration**
- **Maintains healthy group dynamics**
- **Encourages creativity and risk-taking**
- **Recognizes and celebrates team member contributions**

Other Team Roles – Members Can Formally or Informally Take on These Roles

Initiator - Someone who suggests new ideas. One or more people can have this role at a time.

Recorder - This person records whatever ideas a team member may have. It is important that this person quote a team member accurately and not "edit" or evaluate them.

Devil's Advocate/Skeptic - This is someone whose responsibility is to look for potential flaws in an idea.

Optimist - This is someone who tries to maintain a positive frame of mind and facilitates the search for solutions.

Timekeeper - Someone who tracks time spent on each portion of the meeting.

Gate Keeper - This person works to ensure that each member gives input on an issue. One strategy to do this is to ask everyone to voice their opinion one at a time. Another is to cast votes.

Summarizer - Someone who summarizes a list of options.

From Individuals A Group Forms

Help members understand each other

Myers-Briggs Type Indicator (MBTI)

Extraverts ---------- Introverts

Sensors ---------- iNtuitive

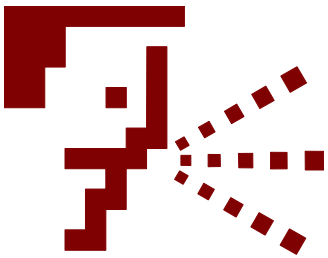
Thinker ---------- Feelers

Judger ---------- Perceiver

By selecting one from each category, we define our personality type, ESTJ, ENTJ…INFP

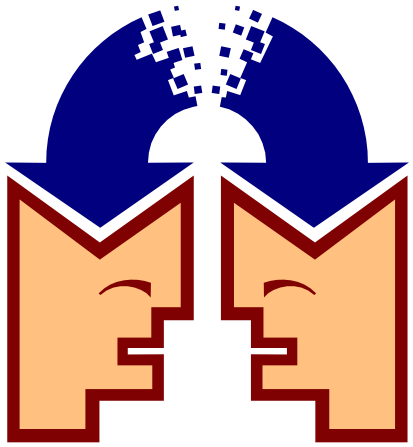
Relevance to Teams (E/I)

- Extraverts
  - Need to think aloud
  - Great explainers
  - May overwhelm others
- Introverts
  - Need time to process
  - Great concentration
  - May not be heard



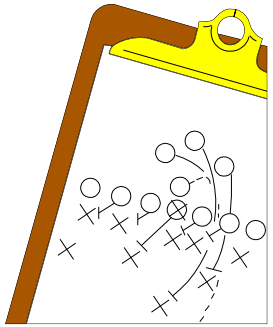
Relevance to Teams (N/S)

- iNtuitive
  - Great at big picture
  - See connections
  - May make mistakes in carrying out plans
- Sensor
  - Great executors
  - May miss big picture, relative importance



Relevance to Teams (T/F)

- Thinker
  - Skillful at understanding how anything works
- Feeler
  - Knows why something matters



Relevance to Teams (J/P)

- Judger
  - Good at schedules, plans, completion
  - Makes decisions easily (quickly)
  - May overlook vital issues
- Perceiver
  - Always curious, wants more knowledge
  - May not get around to acting

What Type are You?

Online Personality Tests

- Jung types http://www.humanmetrics.com/cgi-win/JTypes1.htm

- Keirsey types http://www.keirsey.com/cgi-in/keirsey/newkts.cgi

Stage 2: STORMING

During the ***Storming*** stage team members:

- **realize that the task is more difficult than they imagined**
- **have fluctuations in attitude about chances of success**
- **may be resistant to the task**
- **have poor collaboration**

Storming Diagnosis

- Do we have common goals and objectives?
- Do we agree on roles and responsibilities?
- Do our task, communication, and decision systems work?
- Do we have adequate interpersonal skills?

Negotiating Conflict

- Separate problem issues from people issues.
- Be soft on people, hard on problem.
- Look for underlying needs, goals of each party rather than specific solutions.

Addressing the Problem

- State your views in clear non-judgmental language.
- Clarify the core issues.
- Listen carefully to each person's point of view.
- Check understanding by restating the core issues.

Stage 3: NORMING

- During this stage members accept:
  - their team
  - team rules and procedures
  - their roles in the team
  - the individuality of fellow members

- Team members realize that they are not going to crash-and-burn and start helping each other.

Behaviors

- Competitive relationships become more cooperative.
- There is a willingness to confront issues and solve problems.
- Teams develop the ability to express criticism constructively.
- There is a sense of team spirit.

Giving Constructive Feedback

- Be descriptive
- Don't use labels
- Don’t exaggerate
- Don’t be judgmental
- Speak for yourself

Giving Constructive Feedback

- Use “I” messages.
- Restrict your feedback to things you know for certain.
- Help people hear and accept your compliments when giving positive feedback.

Receiving Feedback

- Listen carefully.
- Ask questions for clarity.
- Acknowledge the feedback.
- Acknowledge the valid points.
- Take time to sort out what you heard.

Stage 4: PERFORMING

Team members have:

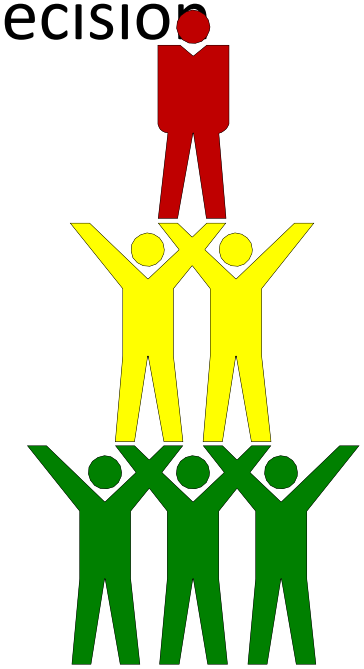
- ✓gained insight into personal and team processes
- ✓a better understanding of each other's strengths and weaknesses
- ✓gained the ability to prevent or work through group conflict and resolve differences
- ✓developed a close attachment to the team

Recipe for Successful Team

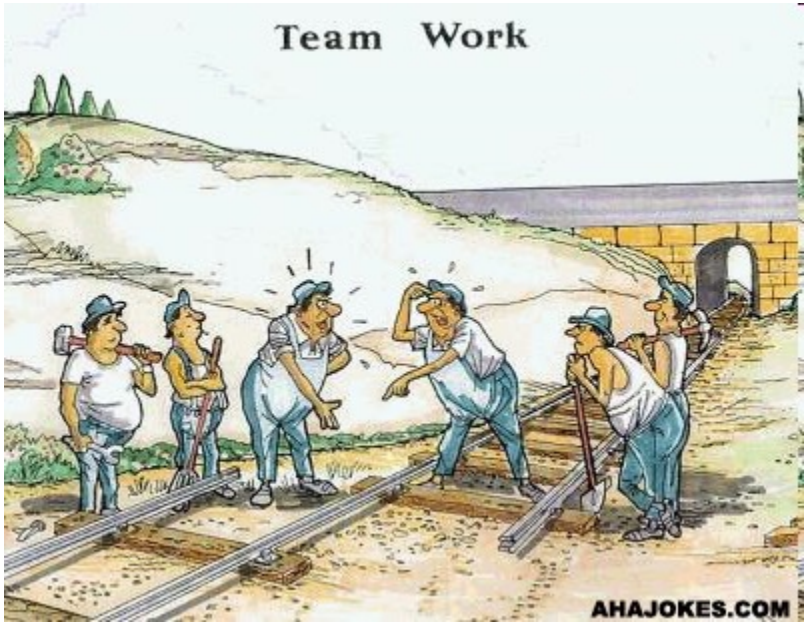
- Commitment to shared goals and objectives
- Clearly define roles and responsibilities
  - Use best skills of each
  - Allows each to develop in all areas

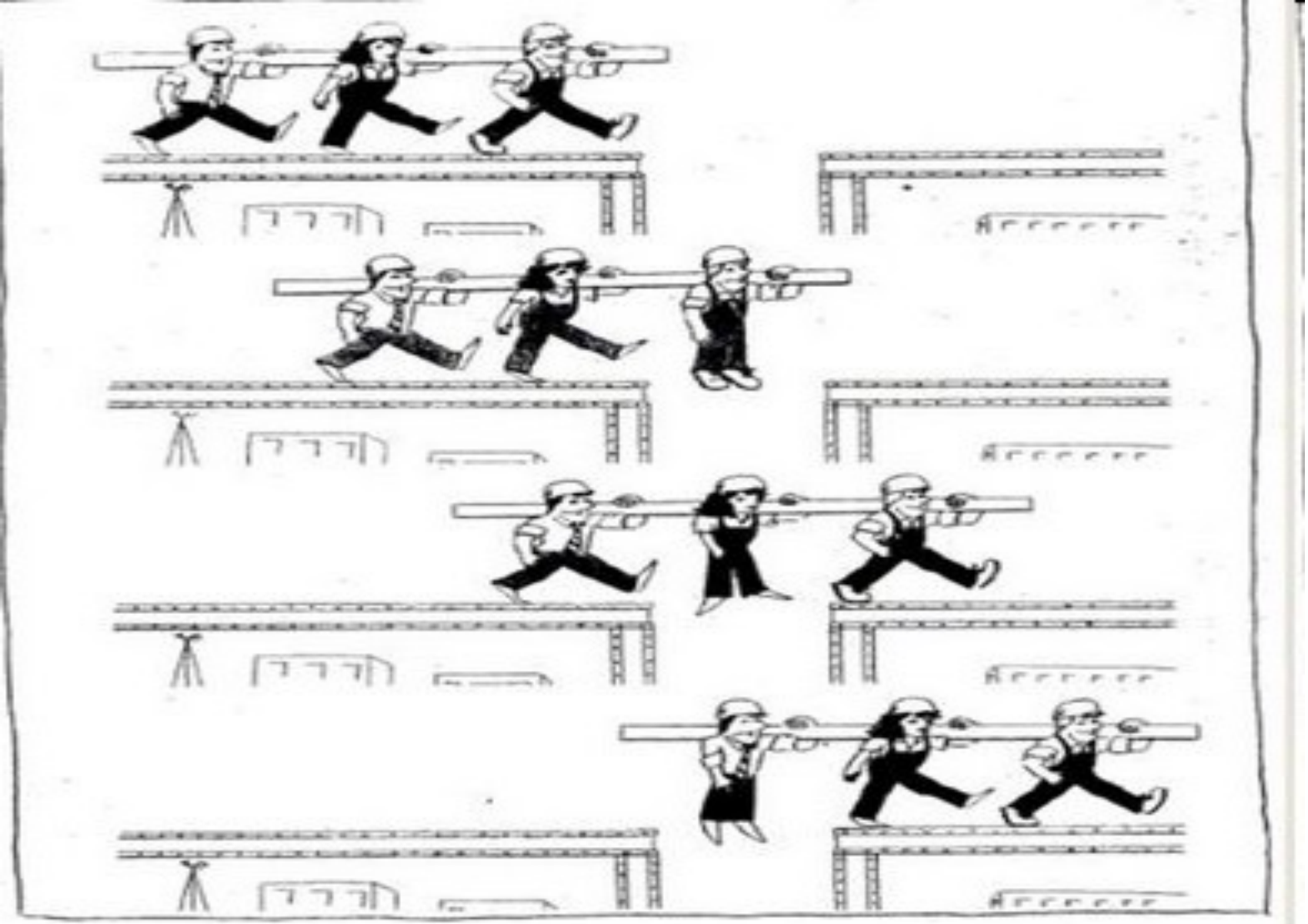
Recipe for Successful Team

- Effective systems and processes
  - Clear communication
  - Beneficial team behaviors; well-defined decision procedures and ground rules
  - Balanced participation
  - Awareness of the group process
  - Good personal relationships



The Results of Team Work





Every member in a Team, has times when they need support - - > > >

References

- Bob Mendonsa and Associates http://www.trainingplus.com
- http://www/unitar.org
- *www.challenge.nm.org/resources/**Team_Building.ppt***

Tutorial

Group Element

- An **interorganizational system** (**IOS**) is a system between organizations, or "shared information system among a group of companies."[For example, an e-commerce supply chain may involve many parties such as a manufacturer/supplier (a company producing products and put them on sale via an online retailer platform), an online retailer (like amazon.com which sells many products produced by different companies) and a logistic company (delivering the products ordered on online retailer platform). Another example is trade finance, in which there is a buyer and seller for goods and services with intermediaries (such as banks which facilitates the transaction by financing the trade).

Group Element (Cont'd)

- Each group may select any interorganizational system and pick two different roles (e.g. manufacturer/retailer/logistics in e-commerce supply chain) and develop a sample mini system to be used in the daily operations of each company. Please note that each mini system used in individual company is an independent system and should not share a central database with other company. You are free to develop your ideas on the necessary functions and objectives of such mini system.
- One requirement is that an interface has to be built with another relevant party for collaboration within the interorganizational system. You are encouraged to use web services/RESTful API in the project. For example, you may call some web services available from cloud provider such as Google for bar code detection/image recognition or open-API provided by banks for payment settlement. You may also build node.js services for interchange of information using JSON string between different parties (e.g. an electronic order can be sent from the online retailer to the manufacturer).

Group Element
- Interim Submission(20%)
- around mid Feb 2023

Your Group will be required to

- conduct an interim presentation relating to service blueprint, Agile project planning, user stories and data interface. The role of how web services/API and prototyping contribute to Agile development should be discussed.
- Conduct a preliminary demonstration of the system prototype (for example, only screen flow without detailed logic being built) and web services/API used.
- Conduct a preliminary demonstration of data exchange

Others

- **Group Element - Final Submission (30%)**
  - For final submission, you will work with the other members of the group to record an 8 min screencast where you demonstrate the product's functionality
- **Individual Element (50%)**
  - You are expected to write your own, individual reflective report on the group project as mentioned above without collusion with other members of your group.
  - You are expected to demonstrate in class the programming exercise on joint tax assessment, the designed test cases and the unit test framework methodology. You are also expected to write your reflection experience on such pair programming.

Report	< 40%	>=40%, <50%	>=50%, <60%	>=60%. <70%	>=70%
Background & Team Dynamics (10%)	Inadequate description of interorganizational system; Poor understanding of team dynamics, roles. and the role of communication within the immediate team.	Basic understanding of interorganizational system and how team decisions and communication impacted on the project.	A clear understanding of the business flow in an interorganizational system and how team decisions and communication impacted the efficiency of the project.	Detailed understanding of interorganizational system including business flow and technical structure; good understanding of how the larger team dynamics and communication impacted on the development process.	Detailed analysis of interorganizational system from business and technical perspective and demonstration of how team dynamics affected the development process and the communication strategy employed.
Version Control (5%)	Flawed understanding of the role of version control.	Basic understanding of how version control helps to improve code quality.	A clear understanding of the key features of version control tools	Detailed understanding of how a range of Git features have supported distributed code development.	Detailed analysis and reflection on how advanced Git tools helped distributed development and improved code quality
Evaluation of Various Agile Development Methodology (10%)	Limited understanding and application of agile methodologies.	Demonstrate understanding and application of basic agile approaches.	Demonstrate an understanding of how agile methodologies and documentation (such as sprint logs) are used to improve software quality and the various agile methodology available	Detailed understanding of how agile methodologies and documentation (such as sprint logs) were used to support work across multiple teams and detailed evaluation of various Agile methodologies when applied to real world solutions.	In-depth analysis and evaluation of a wide range of agile methodologies and documentation (such as sprint logs) applied to large team collaboration supported by case studies.
Evaluation of Various Agile Development tools and web services/RESTful API (10%)	Limited understanding and application of Agile tools and web services	Demonstrate understanding and application of basic agile development tools and web services	Demonstrate an understanding of how agile development tools contributes to development and project management of Agile projects and how web services/RESTful API enhance the agility of developed system	Detailed understanding of how agile development tools and web services/RESTFUL API can be applied to develop real life solutions.	In-depth analysis and evaluation of a wide range of Agile development tools available in the industry (such as Microsoft Agile Development platform) and web services/RESTFUL API
Agile Testing Methodologies 10%	Flawed explanation of various Agile testing techniques.	Basic understanding of how testing improves code quality and how automated testing operates.	A clear understanding of testing stages/methodologies and the key features of automated testing and how the software quality can be improved by unit/system/integration testing.	Detailed understanding of the Agile testing methodology how continuous integration supports agile development and evaluation of various Agile testing tools available in the market.	Detailed analysis showing how continuous integration has supported the development process and in-depth evaluation of various Agile testing tools available in the market.
Referencing 5%	Limited references which are not used in the body of the report or not using CU Harvard.	A limited number of appropriate references that are used in the body of the report.	A range of appropriate references that are used in the report.	A wide range of well-researched references that form a key part of the reflection in the report.	A wide range of well-researched references used a part of a detailed critique within the report.
Product	< 40%	>=40%, <50%	>=50%, <60%	>=60%. <70%	>=70%
The Interim Product (Group) 20%	Simple product that does not demonstrate screen flow.	Basic prototype with incomplete workflow and interfaces	An adequate prototype with a main workflow and key fields	A workable prototype with key fields/features; Basic external interfaces and API call being established	A comprehensive prototype with full workflow demonstrated and successful demonstration of external interfaces and API calls.
The Final Product (GROUP) 30%	Simple product that does not implement core functionality.	Simple functional product demonstrated with limited external interfaces.	Product demonstrates a useful range of functionality and workable interface to other teams.	The product demonstrates advanced functionalities and interfaces to the systems developed by other teams.	A professional product that makes full use of functionality and external interfaces.

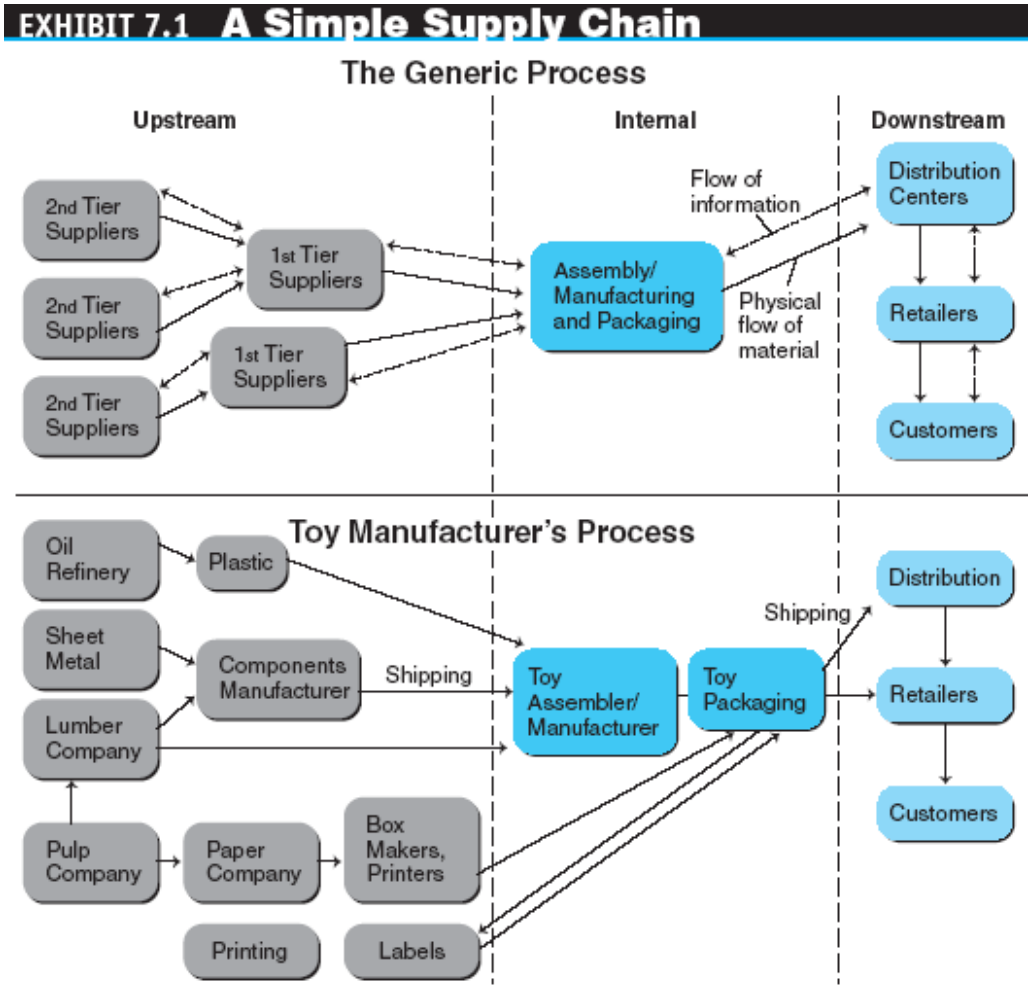
E-Supply Chains

- **Definitions and Concepts**
  - **supply chain**

    The flow of materials, information, money, and services from raw material suppliers through factories and warehouses to the end customers
  - **e-supply chain**

    A supply chain that is managed electronically, usually with Web technologies

E-Supply Chains



E-Supply Chains

- **supply chain management (SCM)**

  A complex process that requires the coordination of many activities so that the shipment of goods and services from supplier right through to customer is done efficiently and effectively for all parties concerned. SCM aims to minimize inventory levels, optimize production and increase throughput, decrease manufacturing time, optimize logistics and distribution, streamline order fulfillment, and overall reduce the costs associated with these activities

E-Supply Chains

- **e-supply chain management (e-SCM)**

  The collaborative use of technology to improve the operations of supply chain activities as well as the management of supply chains
- The success of an e-supply chain depends on:
  - The ability of all supply chain partners to view partner collaboration as a strategic asset
  - A well-defined supply chain strategy
  - Information visibility along the entire supply chain
  - Speed, cost, quality, and customer service
  - Integrating the supply chain more tightly

E-Supply Chains

- **Activities and infrastructure of E-SCM**
  - Supply chain replenishment
  - E-procurement
  - Supply chain monitoring and control using RFID
  - Inventory management using wireless devices
  - Collaborative planning
  - Collaborative design and product development
  - E-logistics
  - Use of B2B exchanges and supply webs

E-Supply Chains

- **e-procurement**

  The use of Web-based technology to support the key procurement processes, including requisitioning, sourcing, contracting, ordering, and payment. E-procurement supports the purchase of both direct and indirect materials and employs several Web-based functions such as online catalogs, contracts, purchase orders, and shipping notices

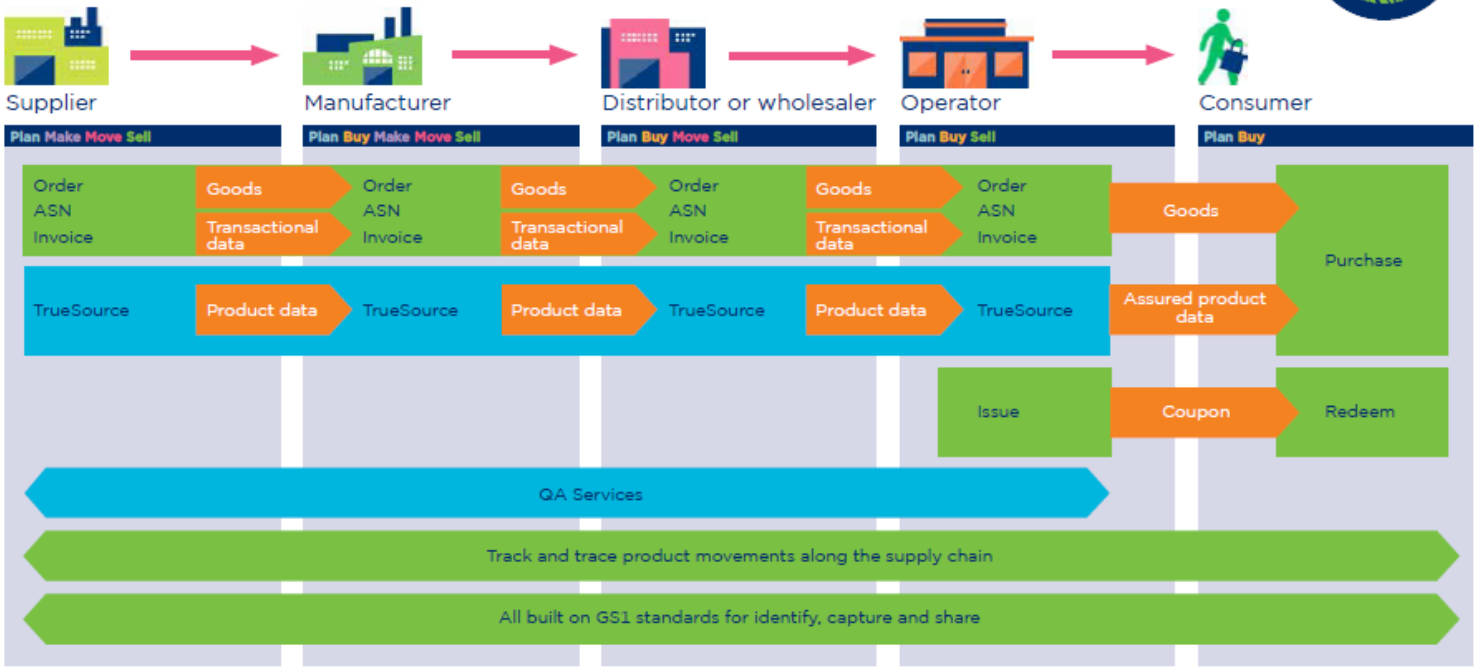
- **collaborative planning**

  A business practice that combines the business knowledge and forecasts of multiple players along a supply chain to improve the planning and fulfillment of customer demand

E-Supply Chains

- **Infrastructure for e-SCM**
  - Electronic Data Interchange (EDI)
  - Extranets
  - Intranets
  - Corporate portals
  - Workflow systems and tools
  - Groupware and other collaborative tools

The foodservice supply chain



Credit Card Transaction Flow

Credit Card Transaction Flow

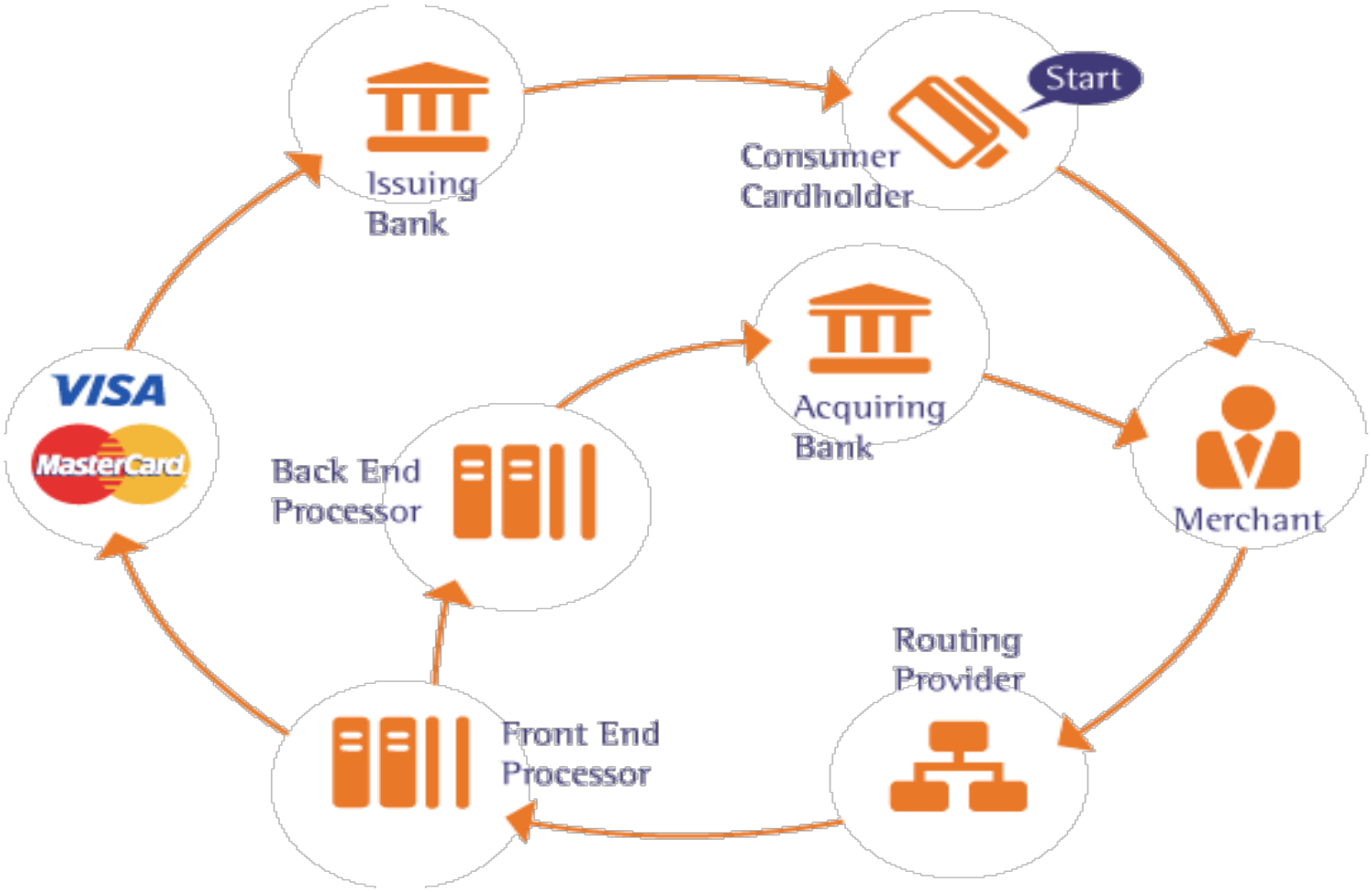
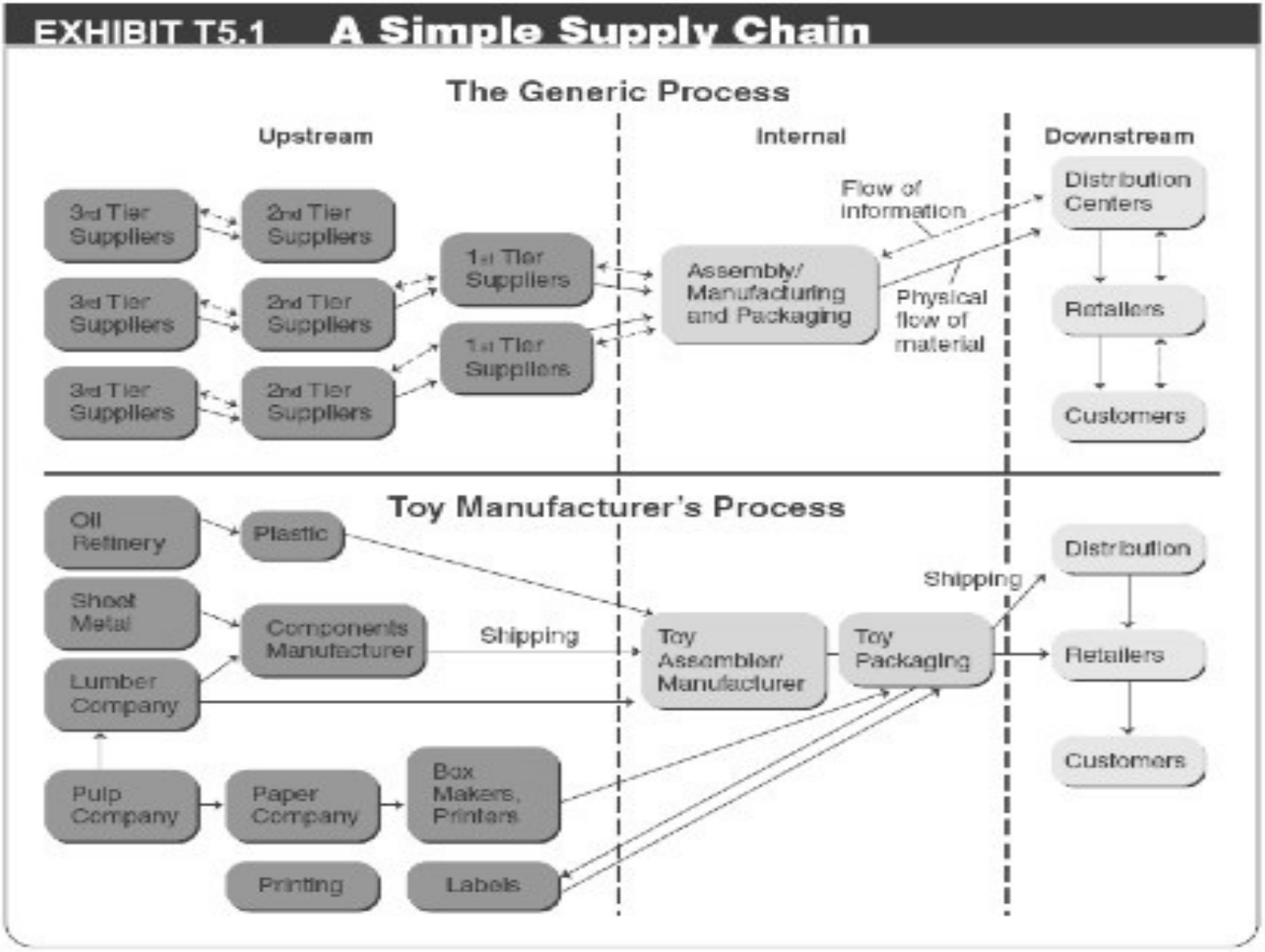
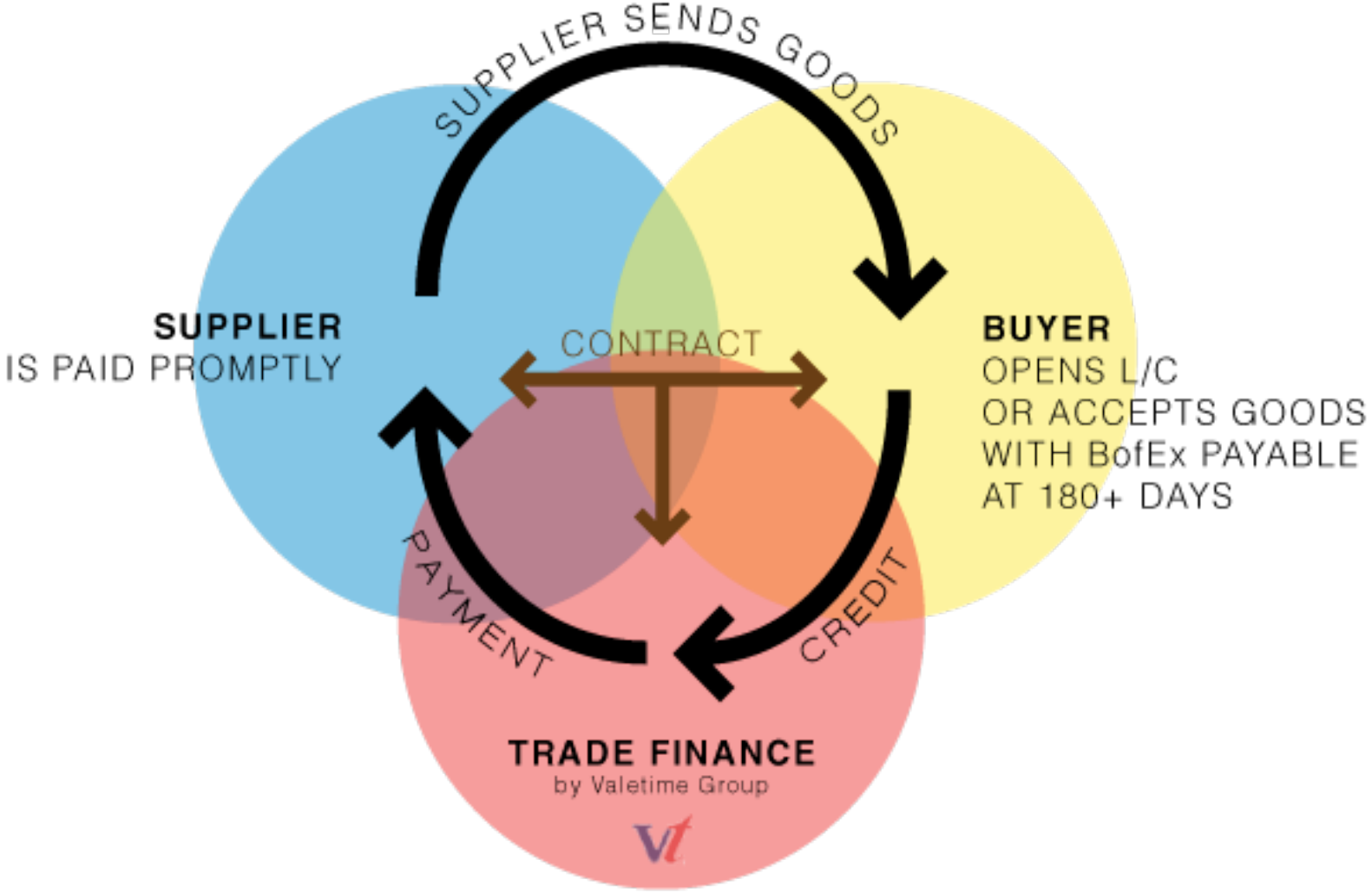


EXHIBIT T5.1 A Simple Supply Chain

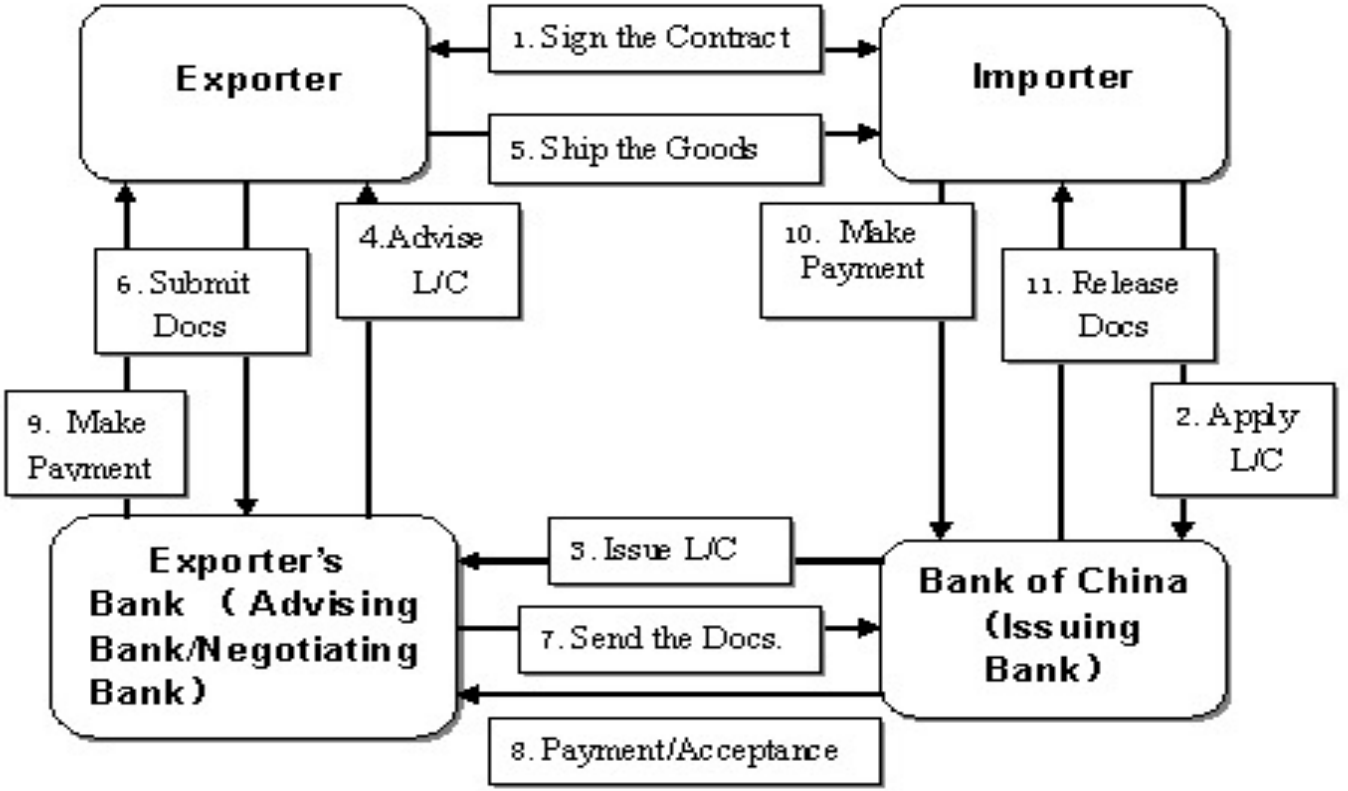


Source: Drawn by E. Turban.

Trade Finance



Trade Finance



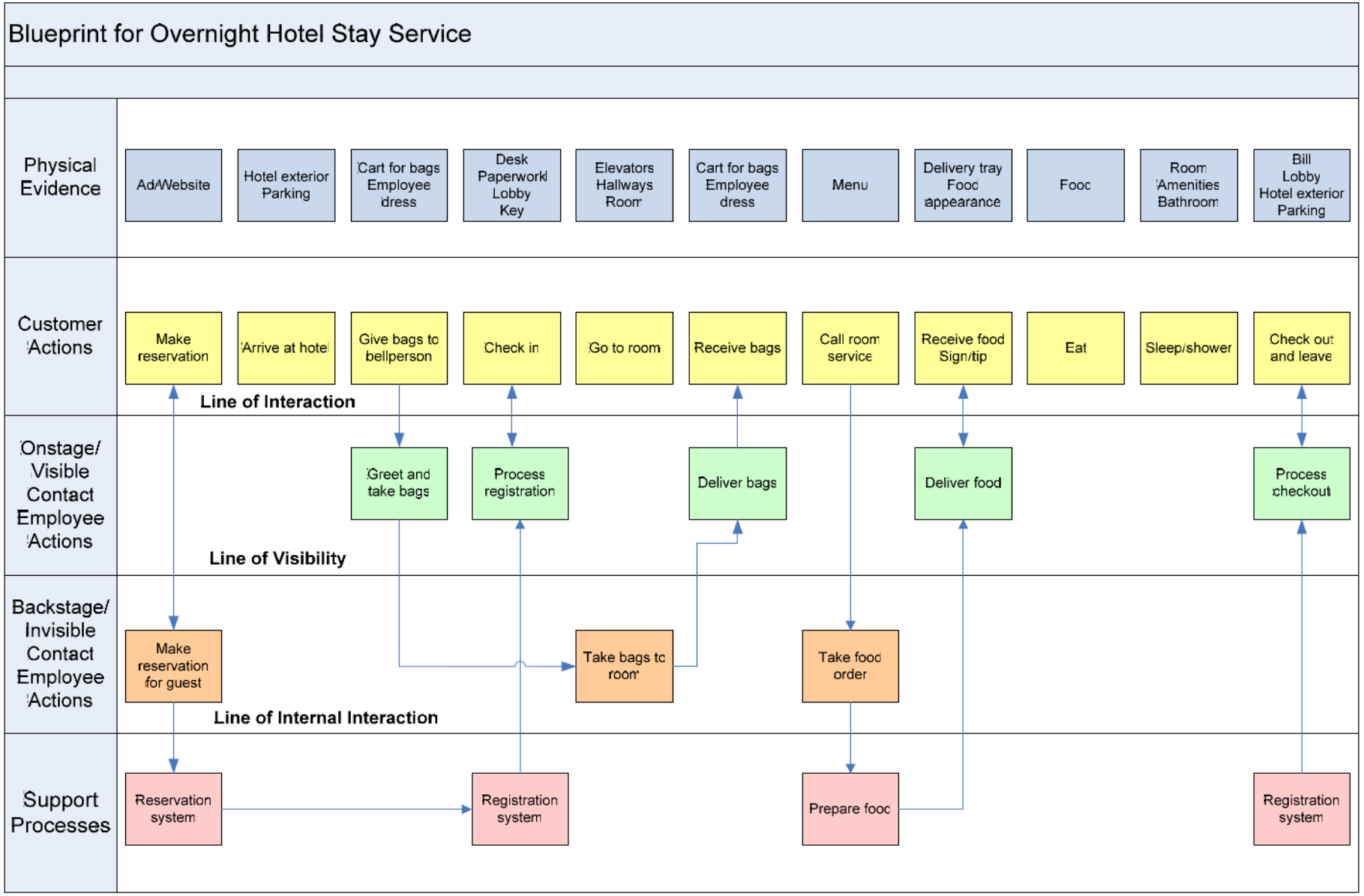
Service Blueprint

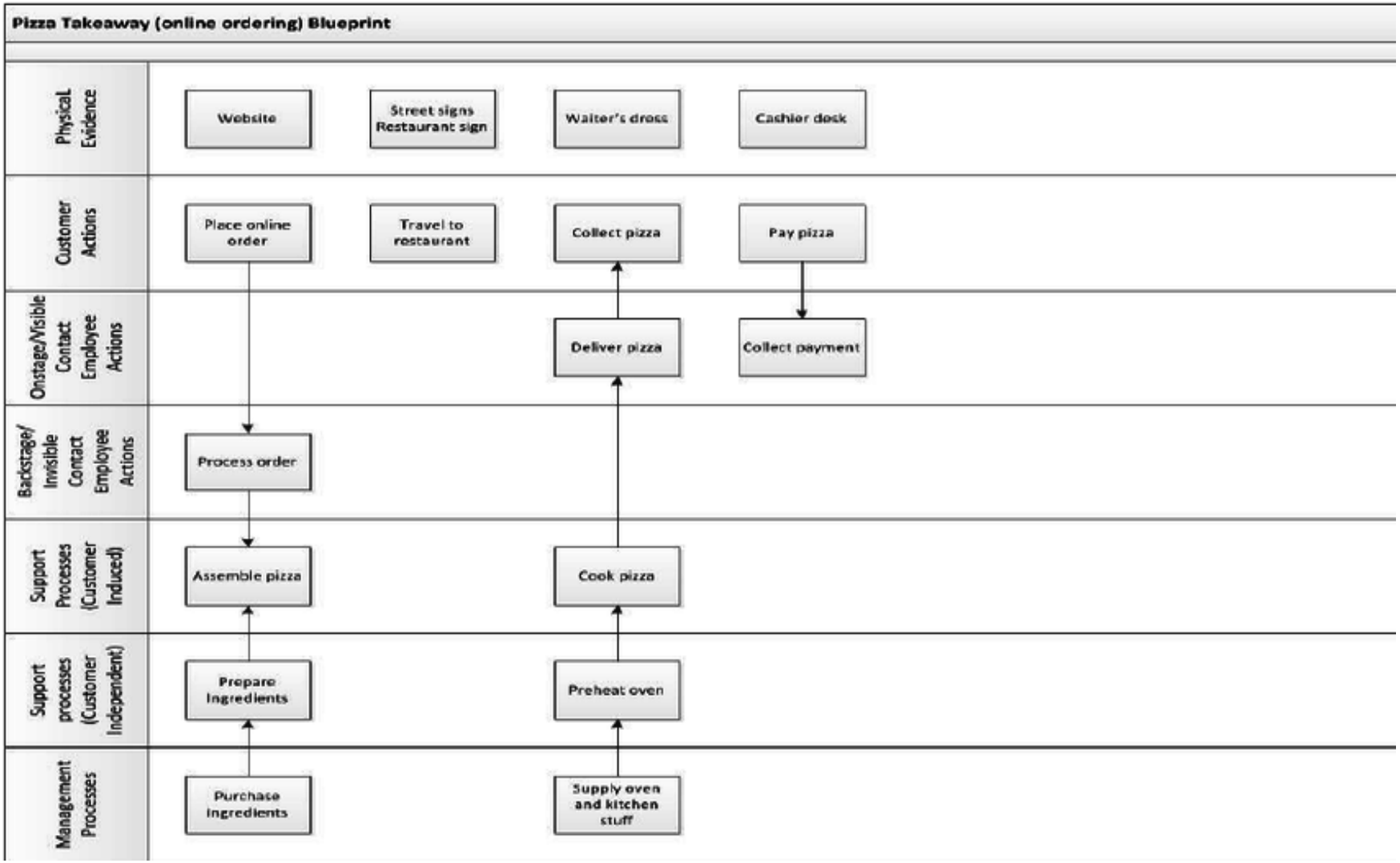
- A service blueprint is an operational planning tool that provides guidance on how a service will be provided, specifying the physical evidence, staff actions, and support systems / infrastructure needed to deliver the service across its different channels.

Elements of Service Blueprint

A typical service blueprint identifies:

- Customer Actions: The steps that customers take as part of the service delivery process.
- Front-stage (Visible Contact Employee) Actions: Steps taken by contact employees as part of the face-to-face service encounter.
- Back-stage (Invisible Contact Employee) Actions: (The 'line of visibility' separates the front-stage and back-stage actions). Non-visible steps taken by contact employees behind the line of visibility. e.g. taking a hotel or restaurant reservation by telephone.
- Support Processes: Activities carried out by employees who are not contact employees, but whose actions are required for the service to be delivered.
- Physical Evidence: Tangible elements associated with each step that has the potential to influence customer perceptions of the service encounter e.g. uniforms, delivery vans
- Line of Visibility: Line that separates front-stage and back-stage actions


Blueprint for Overnight Hotel Stay Service
Physical Evidence
Ad/Website
Hotel exterior Parking
Cart for bags Employee dress
Desk Paperwork Lobby Key
Elevators Hallways Room
Cart for bags Employee dress
Menu
Delivery tray Food appearance
Fooc
Room Amenities Bathroom
Bill Lobby Hotel exterior Parking
Customer Actions
Make reservation
Arrive at hotel
Give bags to bellperson
Check in
Go to room
Receive bags
Call room service
Receive food Sign/tip
Eat
Sleep/shower
Check out and leave
Line of Interaction
Onstage/ Visible Contact Employee Actions
Greet and take bags
Process registration
Deliver bags
Deliver food
Process checkout
Line of Visibility
Backstage/ Invisible Contact Employee Actions
Make reservation for guest
Take bags to room
Take food order
Line of Internal Interaction
Support Processes
Reservation system
Registration system
Prepare fooc
Registration system


Pizza Takeaway (online ordering) Blueprint
Physical Evidence
Website
Street signs Restaurant sign
Waiter's dress
Cashier desk
Customer Actions
Place online order
Travel to restaurant
Collect pizza
Pay pizza
Onstage/Visible Contact Employee Actions
Deliver pizza
Collect payment
Backstage/ Invisible Contact Employee Actions
Process order
Support Processes (Customer Induced)
Assemble pizza
Cook pizza
Support processes (Customer Independent)
Prepare Ingredients
Preheat oven
Management Processes
Purchase ingredients
Supply oven and kitchen stuff

Activity Diagram

Activity Diagram

- Activity diagrams describe the workflow behavior of a system.
  - Activity diagrams are used in process modeling and analysis of during requirements engineering.
  - A typical business process which synchronizes several external incoming events can be represented by activity diagrams.

- They are most useful for understanding work flow analysis of synchronous behaviors across a process.

Activity Diagram

- Activity diagrams are used for
  - documenting existing process
  - analyzing new Process Concepts
  - finding reengineering opportunities.

- The diagrams describe the state of activities by showing the sequence of activities performed.
  - they can show activities that are conditional or parallel.

Activity Diagram Concepts

- An activity is trigged by one or more events and activity may result in one or more events that may trigger other activity or processes.
- Events start from start symbol and end with finish marker having activities in between connected by events.
- The activity diagram represents the decisions, iterations and parallel/random behavior of the processing.
  - They capture actions performed.
  - They stress on work performed in operations (methods).

When to Use Activity Diagrams

- The main reason to use activity diagrams is to model the workflow behind the system being designed.
- Activity Diagrams are also useful for:
  - analyzing a use case by describing what actions need to take place and when they should occur
  - describing a complicated sequential algorithm
  - modeling applications with parallel processes
- Activity Diagrams should not take the place of interaction diagrams and state diagrams.
- Activity diagrams do not give detail about how objects behave or how objects collaborate.

Components

- An *activity* is an ongoing, though interruptible, execution of a step in a workflow (such as an operation or transaction)
  - Represented with a rounded rectangle.
  - Text in the activity box should represent an activity (verb phrase in present tense).

Components

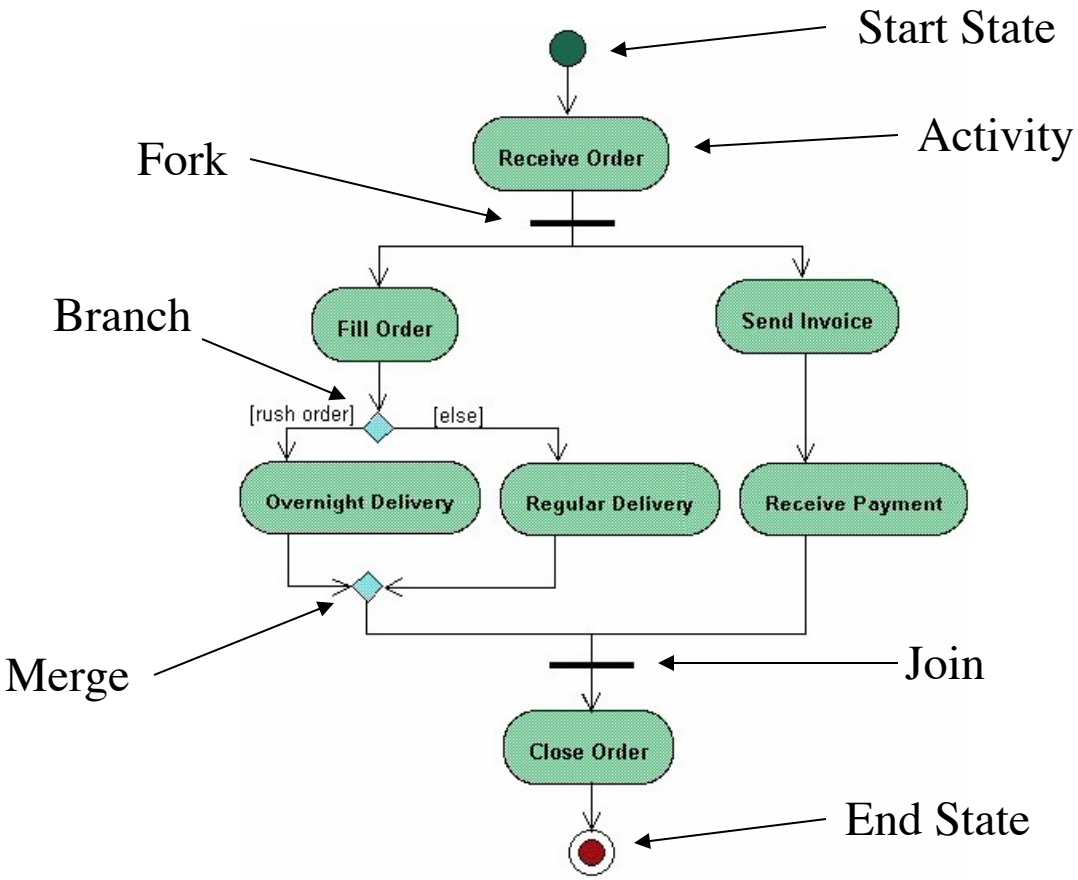
- An *event* is triggered by an activity. It specifies a significant occurrence that has a location in time and space.
  - An instance of an event (trigger) results in the flow from one activity to another.
  - These are represented by directed straight lines emerging from triggering activity and ending at activity to be triggered. Label text for events should represent event but not the data involved.

- A *decision* may be shown by labeling multiple output transitions of an activity with different guard conditions.
  - For convenience a stereotype is provided for a decision: the traditional diamond shape, with one or more incoming arrows and with two or more outgoing arrows, each labeled by a distinct guard condition with no event trigger.

How to Draw an Activity Diagram

- Diagrams are read from top to bottom and have branches and forks to describe conditions and parallel activities.
  - A fork is used when multiple activities are occurring at the same time.
  - A branch describes what activities will take place based on a set of conditions.
  - All branches at some point are followed by a merge to indicate the end of the conditional behavior started by that branch.
  - After the merge all of the parallel activities must be combined by a join before transitioning into the final activity state.

Activity Diagram Example



Disadvantages

- A disadvantage of activity diagrams is that they do not explicitly present which objects execute which activities, and the way that the messaging works between them.
  - Labeling of each activity with the responsible object can be done.
  - It is useful to draw an activity diagram early in the modeling of a process, to help understand the overall process.

- Then interaction diagrams can be used to help you allocate activities to classes.

References

- Activity Diagrams
- http://pigseye.kennesaw.edu/~dbraun/csis4650/A&D/UML_tutorial/activity.htm
- http://isds.bus.lsu.edu/cvoc/learn/bpr/cprojects/spring1998/modeling/activity.html
- http://www-106.ibm.com/developerworks/rational/library/2802.html
- Fast Track UML 2 (from Books 24x7)

Examples of UML Diagram

https://www.uml-diagrams.org/index-examples.html

Class Exercise

- Present it in class the two or more different types of companies participating in a supply chain(relating to your assignment). The supply chain can be about some scenarios based on your company , food production and delivery, health service or etc.
- Draw a high level of activity diagram showing the workflow of the interface between the two companies
- At the end of class, take a photo of what you have drawn (or share your softcopy), the hardcopy/softcopy should have the list of your team members with full names, upload it to Canvas and present in the class
- For later assignment submission posted on Canvas, you may use www.draw.io to draw the charts