EFOP-3.4.3-16-2016-00009

A felsőfokú oktatás minőségének és hozzáférhetőségének együttes javítása a Pannon Egyetemen

SYSTEM ANALYSIS AND SYSTEM DESIGN

Imre Budai

Based on:

Dennis-Wixom-Roth: System Analysis & Design

5th edition

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

EFOP-3.4.3-16-2016-00009

A felsőfokú oktatás minőségének és hozzáférhetőségének együttes javítása a Pannon Egyetemen

SYSTEM ANALYSIS AND SYSTEM DESIGN

1. ANALYST AND DEVELOPMENT

Imre Budai

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

Objectives

- Explain the role played in information systems development by the systems analyst.
- Describe the fundamental systems development life cycle and its four phases.
- Explain how organizations identify IS development projects.
- Explain the importance of linking the information system to business needs.

Objectives

- Be able to create a system request.
- Describe technical, economic, and organizational feasibility assessment.
- Be able to perform a feasibility analysis.

Introduction

- Systems Development Life Cycle
  - How an information system (IS) can:
    - Support business needs,
    - Designing the system,
    - Building it,
    - Delivering it to users.

Introduction

- An estimated $2.4 trillion was spent by organizations and governments on IT hardware, software, and services worldwide in 2010.
- It is projected to increase by 3.5% in 2011.
- Success is “improbable” in 68% of technology projects.
  - Totally abandoned, or
  - Delivered to the users significantly late, cost far more than expected, and have fewer features than originally planned.
- The key person is the systems analyst.

Introduction

- A software error resulted in Toys R Us double billing some shoppers for purchases made on Black Friday.
- Verizon Wireless had to refund $50 million to customers due to billing system errors.
- Chase banking customers were unable to access their online banking accounts for over 24 hours due to a computer glitch.
- McAfee’s anti-virus software product caused its users’ computers to lock up. McAfee offered affected customers a free 2-year subscription and reimbursement for costs incurred to repair the machines.

Introduction

- The key person in the SDLC is the systems analyst.
  - Analyses the business situation.
  - Identifies opportunities for improvements.
  - Designs an information system to implement the improvements.
- The primary objective of the systems analyst is not to create a wonderful system, but:
  - To create value for the organization,
  - Which for most companies means increasing profits.

Introduction

- Main reasons for failures are because the analysts tried to build a wonderful system without clearly understanding
  - How the system would support the organization’s goals.
  - How to improve business processes.
  - How to integrate with other information systems to provide value.

- The goal is to enable the organization to perform work better so that it can earn greater profits or serve its constituents more effectively.

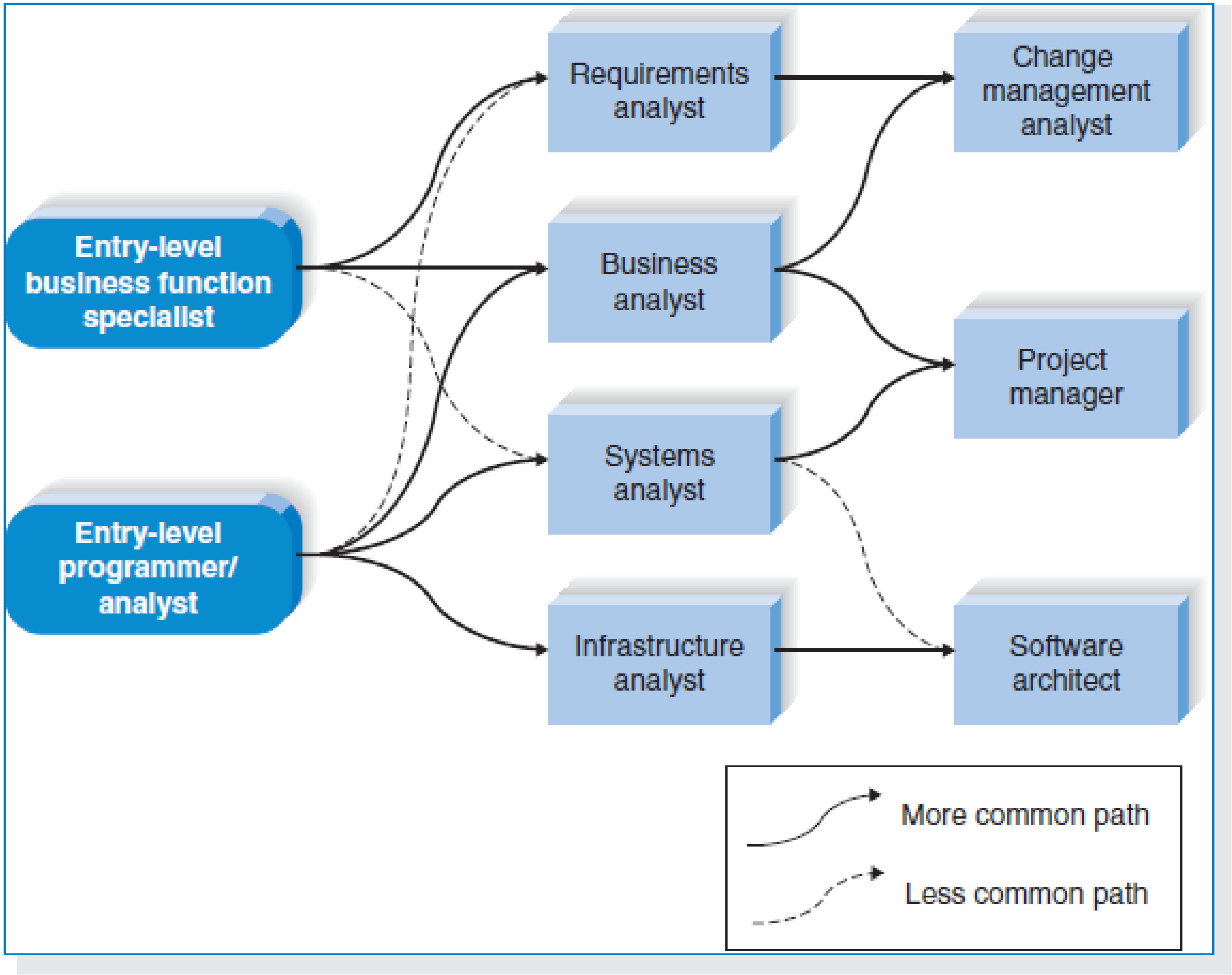
System analyst

- Key role in information systems development projects
- Works closely with all project team members
- May serve as change agents
- Skills: technical, business, analytical, interpersonal, management, and ethical
- Must deal fairly, honestly, and ethically with other project team members, managers, and system users.
- Often deals with confidential information

System analyst

- Roles: focuses on the business issues surrounding the system.
  - Requirements analyst: eliciting the requirements from the stakeholders
  - Infrastructure analyst: technical issues
  - Change management analyst: people and management issues
  - Project manager: ensures that the project is completed on time and within budget and that the system delivers the expected value to the organization

System analyst



SDLC

- Building an information system is similar to building a house:
    - First, the owner describes the vision.
    - Second, this idea is transformed into sketches and drawings.
    - Third, a set of detailed blueprints is developed.
    - Finally, the house is built following the blueprints - and often with some changes and decisions made by the owner as the house is erected.

- SDLC follows a similar set of four fundamental phases:

  Planning, analysis, design, and implementation.

SDLC

- Planning
  - Project initiation
    - System request
    - Feasibility analysis
      - The technical feasibility (Can we build it?)
      - The economic feasibility (Will it provide business value?)
      - The organizational feasibility (If we build it, will it be used?)
  - Approval committee decides whether the project should be undertaken.

SDLC

- Planning
  - Once the project is approved, it enters project management.
  - Project manager:
    - Creates a work plan;
    - Staffs the project;
    - Puts techniques in place;

      to help the project team control and direct the project through the entire SDLC

SDLC

- Analysis
  - It answers the questions:
    - Who will use the system;
    - What the system will do;
    - Where and when it will be used.
  - project team:
    - Investigates any current system(s);
    - Identifies improvement opportunities;
    - Develops a concept for the new system.

SDLC

- Analysis has three phases:
  - Analysis strategy;
  - Requirements gathering;
  - System proposal.

  - The system proposal is the initial deliverable that describes what business requirements the new system should meet.

SDLC

- Design
  - It decides:
    - How the system will operate in terms of the hardware, software, and network infrastructure;
    - The user interface, forms, and reports;
    - The specific programs, databases, and files.
  - The design strategy.
  - The basic architecture design.
  - The interface design, the forms and reports.
  - The database and file specifications.
  - The program design.

SDLC

- Implementation
  - System construction

    The system is built and tested to ensure that it performs as designed.
  - Installation

    The old system is turned off and the new one is turned on.
  - Support plan
    - Formal or informal post-implementation review
    - A systematic way for identifying major and minor changes needed

Identification and initiation

- Where do project ideas come from?
  - Always an identified business need.
  - From business
    - New opportunities.
    - Improvements.
    - Business losses
  - From technology
    - New, emerging technologies.
    - Optimization
  - From the regulator side

Identification and initiation

- Benefits can be:
    - Enhanced agility, rapid adaption to changing business.
    - Improved process alignment with industry "best practices".
    - Increased process efficiencies.

- BPM general cycle:
    - 1. defining and mapping the steps in a business process,
    - 2. creating ways to improve on steps – that add value,
    - 3. finding ways to eliminate or consolidate steps – that don't add value,
    - 4. creating or adjusting workflows to match the improved process maps.

Identification and initiation

- Improvement types:
  - Business process automation

    Technology components are used to complement or substitute for manual information management processes.
  - Business process improvement

    Creates new, redesigned processes.
  - Business process reengineering

    Means changing the fundamental way in which the organization operates.

Identification and initiation

- Project sponsor: a person (or group) who has an interest in the system's success:
  - Develops the initial vision of the new system.
  - Works throughout the SDLC.

- High-level business requirements
  - Describe the reasons for developing the system;
  - Outline the benefits it will provide.

Identification and initiation

- Business value
  - Tangible value

    Can be quantified and measured easily.
  - Intangible value

    an intuitive belief that the system provides important, but hard-to-measure, benefits

Identification and initiation

- System Request
  - A document that describes:
    - The business reasons for building a system;
    - The value that the system is expected to provide.
  - It includes:
    - Project sponsor;
    - Business need;
    - Business requirements;
    - Business value;
    - Special issues.

Feasibility analysis

- Technical Feasibility
  - Familiarity with application;
  - Familiarity with technology;
  - Project size;
  - Compatibility.

Feasibility analysis

- Economic Feasibility
  - Development costs;
  - Annual operating costs;
  - Annual benefits;
  - Intangible benefits and costs.

Feasibility analysis

- Organizational Feasibility
  - Project champion(s);
  - Senior management;
  - Users;
  - Other stakeholders.

  - Is the project strategically aligned with the business?

THANKS FOR ATTENTION!

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

EFOP-3.4.3-16-2016-00009

A felsőfokú oktatás minőségének és hozzáférhetőségének együttes javítása a Pannon Egyetemen

SYSTEM ANALYSIS AND SYSTEM DESIGN

2. PROJECT SELECTION AND MANAGEMENT

Imre Budai

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

Objectives

- Explain how projects are selected in some organizations.
- Describe various approaches to the SDLC that can be used to structure a development project.
- Explain how to select a project methodology based on project characteristics.
- Become familiar with project estimation.

Objectives

- Be able to create a project work plan.
- Describe project staffing issues and concerns.
- Describe and apply techniques to coordinate and manage the project.
- Explain how to manage risk on the project.

Introduction

- Importance of chief information officers (CIOs) has been increased.
- Focus is on projects that will provide the highest possible return on IT investments.
- Project portfolio management
- Project manager
- Pressure on the project management
- Unrealistic deadlines

Introduction

- Critical success factors are:
  - Appropriate project selection;
  - Realistic assessment;
  - Keeping focus on the original plan.

Project selection

- Investments in information systems:
  - Are evaluated in the context of an entire portfolio of projects;
  - Decision makers:
    - Look beyond project cost;
    - consider a project’s anticipated risks and returns;
    - Prioritize their business strategies;
    - Assemble and assess project portfolios.
  - The focus on a project’s contribution to an entire portfolio.

Project selection

- Considerations are taken into account by portfolio management:
  - Size;
  - Cost;
  - Purpose;
  - Length;
  - Risk;
  - Scope;
  - Economic value.
- The approval committee must be selective about where to allocate resources.

Creating project plan

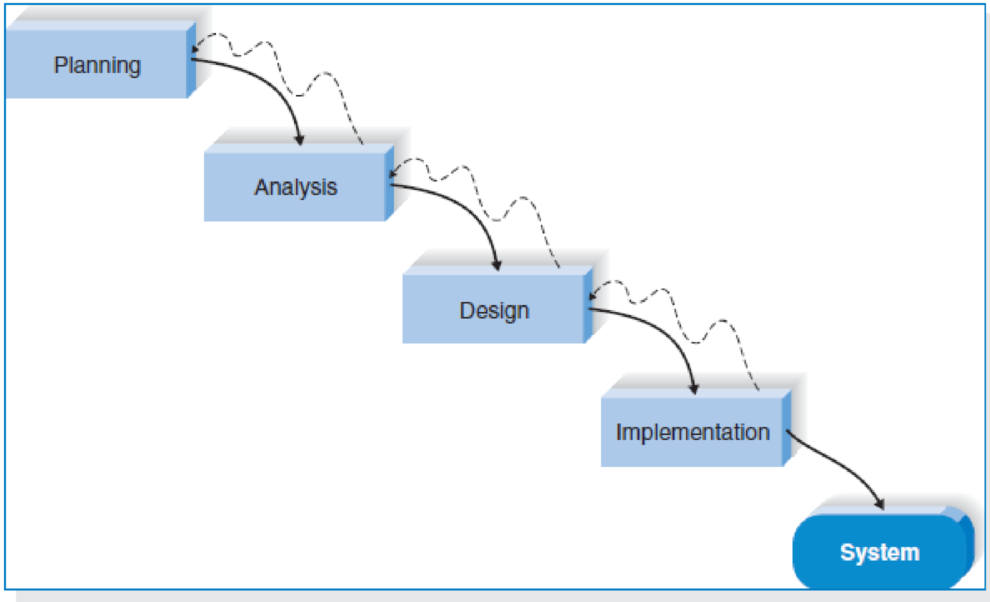
- Project methodology options;
- Case tools;
- Selecting the appropriate development methodology;
- Estimating the project time frame;
- Developing the Work Plan.

Project methodology

- Waterfall Development;
  - Parallel development;
  - V-model;
- Rapid application development;
  - Iterative development;
  - System prototyping;
  - Throwaway prototyping;
- Agile Development;
- Extreme programming

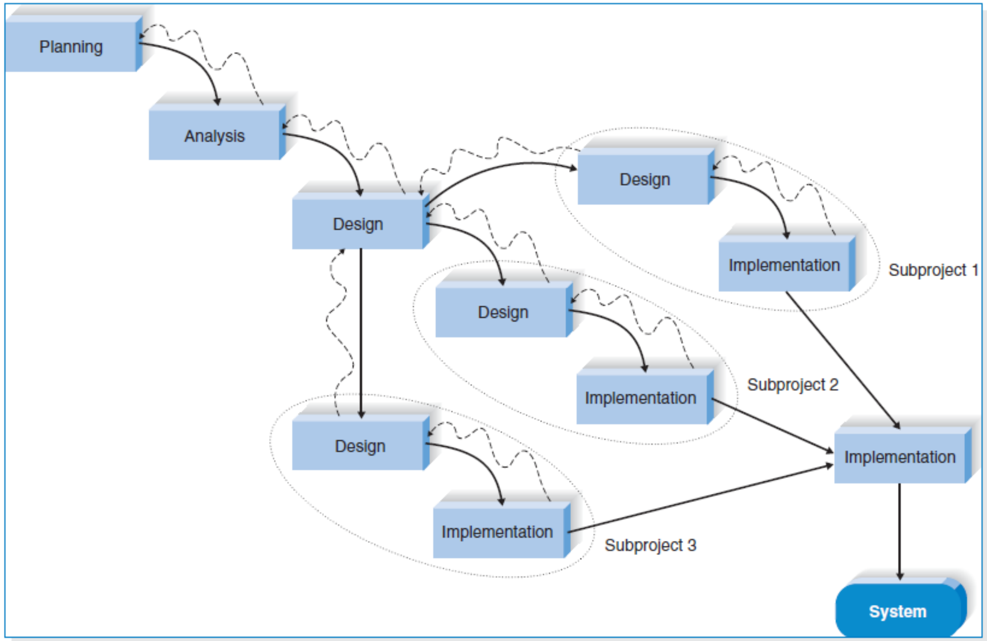
Waterfall development

- Requirements are clear and stable.
- Only limited changes can be possible in requirements.
- Expensive post-implementation.



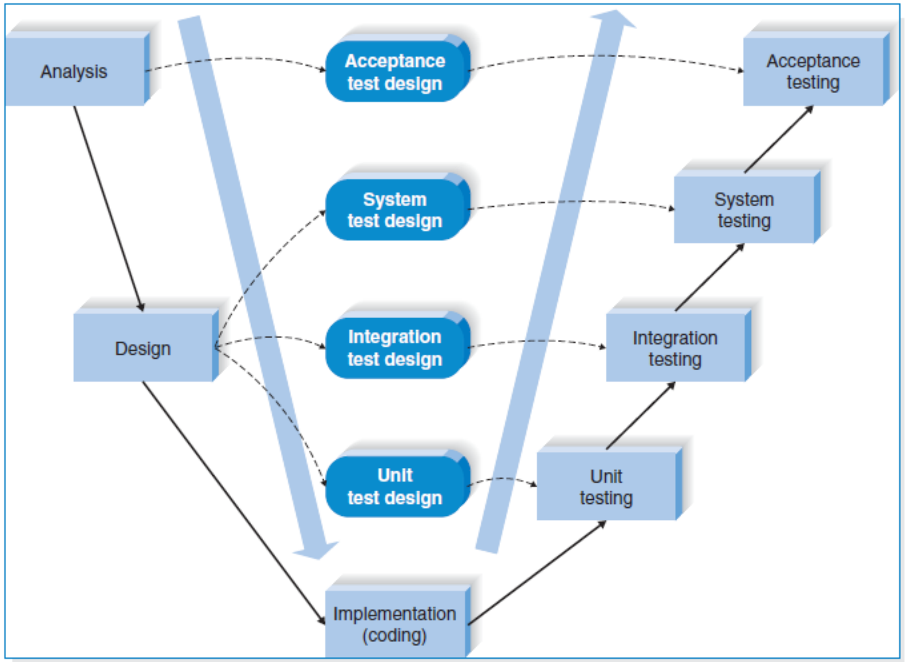
Parallel development

- It is a variant of waterfall development.
- It reduces the time required to deliver a system.
- It has the same limitations and, even more, subprojects are not completely independent.



V-model

- It is another variation of waterfall development.
- Left-hand slope: requirements and design.
- At the base of the V: the code is written.
- Right-hand slope:
    - Testing of components;
    - Integration and testing;
    - Acceptance testing.



Rapid application development

- It is the response to the weaknesses of waterfall development and its variations.
- Special techniques and computer tools:
  - CASE;
  - JAD sessions;
  - Fourth-generation/visual programming languages;
  - code generators.
- It may also introduce a problem in managing user expectations.

Rapid application development

- RAD may be conducted in a variety of ways:
  - Iterative development;

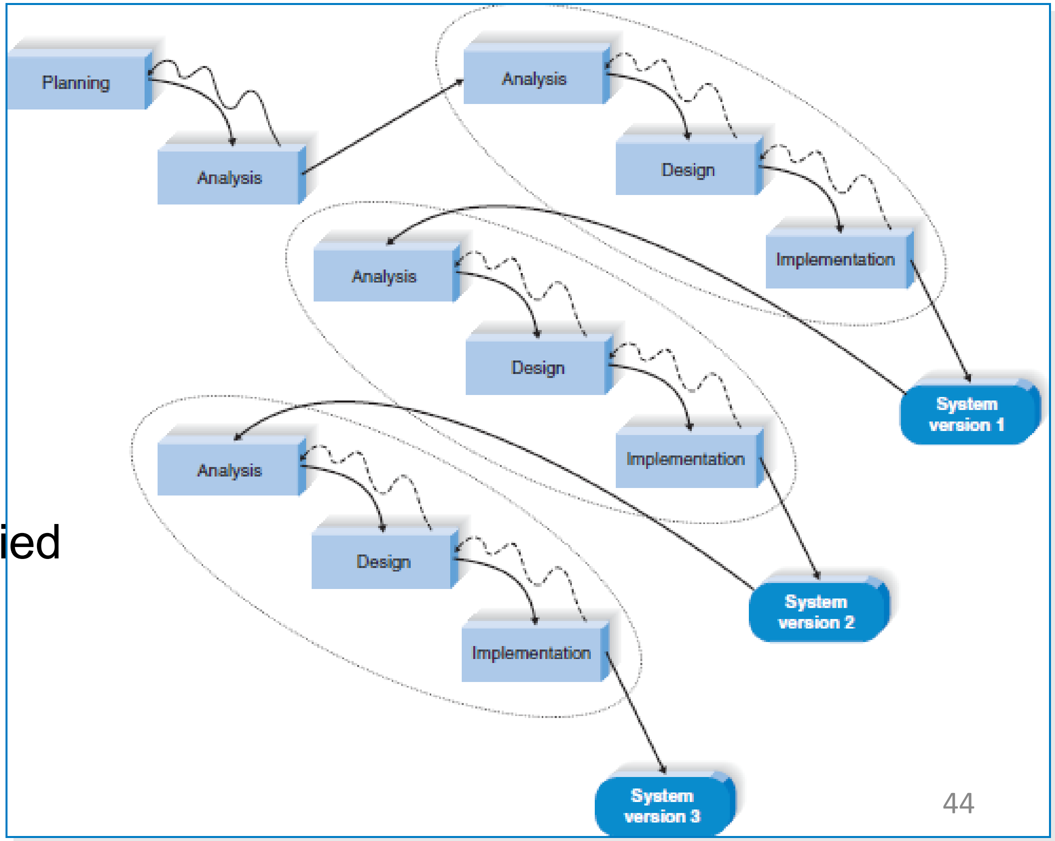
    It breaks the overall project into a series of versions that are developed sequentially.
  - System prototyping;

    It performs the analysis, design, and implementation phases concurrently in order to quickly develop a simplified version of the proposed system and give it to the users for evaluation and feedback.
  - Throwaway prototyping;

    Uses the prototypes primarily to explore design alternatives.

Iterative development

- The most important and fundamental requirements are bundled into the first version of the system.
- Mini-waterfall process.
- Valuable feedback.
- Next version of the system.
- Important additional requirements may be identified



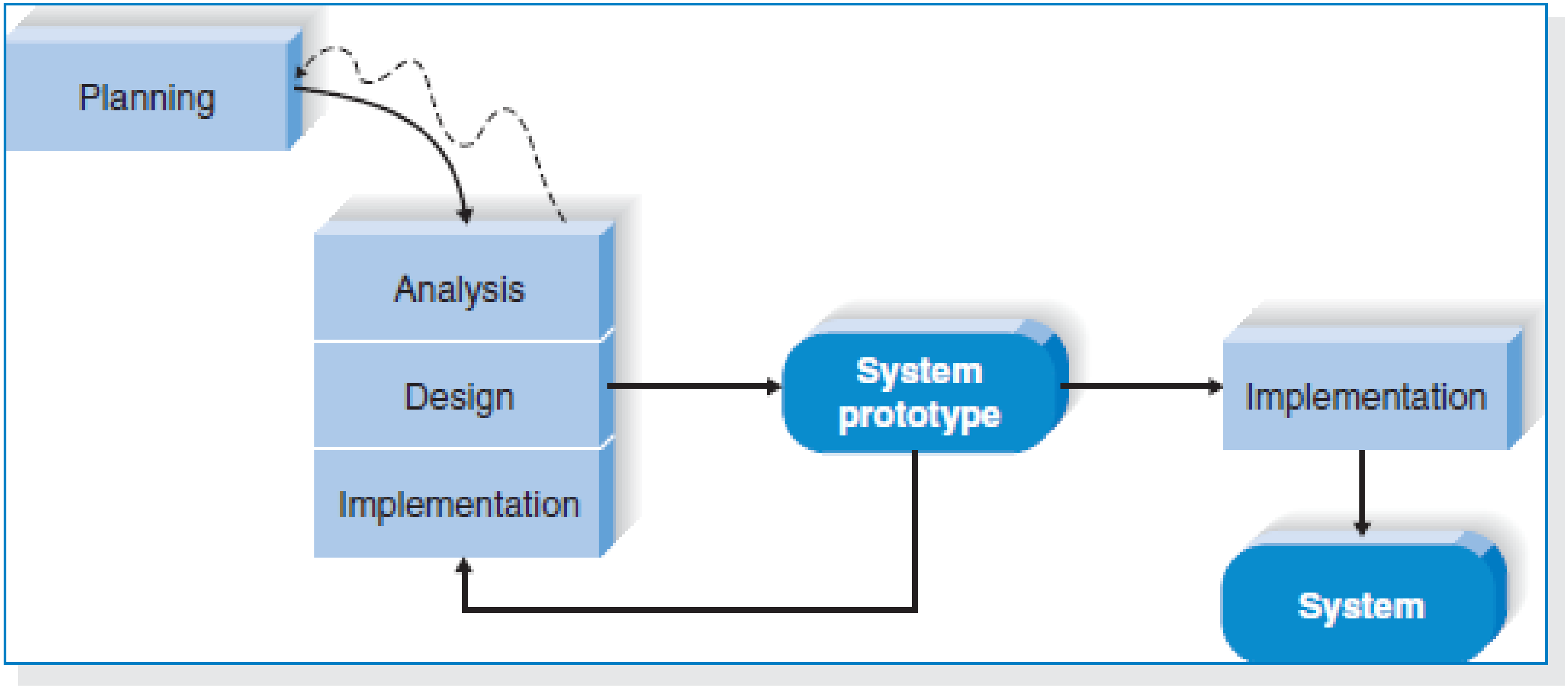
System prototyping

- A “quick and dirty” version of the system.
- provides minimal features.
- Reactions and comments from the users.
- A second/next prototype that corrects deficiencies and adds more features.
- Cycles continue until the analysts, users, and sponsor agree that the prototype provides enough functionality to be installed and used in the organization.

System prototyping

- It provides a system very quickly for users to evaluate and reassures users that progress is being made.
- It is very useful when users have difficulty expressing requirements for the system.
- A disadvantage, is the lack of careful, methodical analysis prior to making design and implementation decisions.
- It may have some fundamental design limitations.

System prototyping

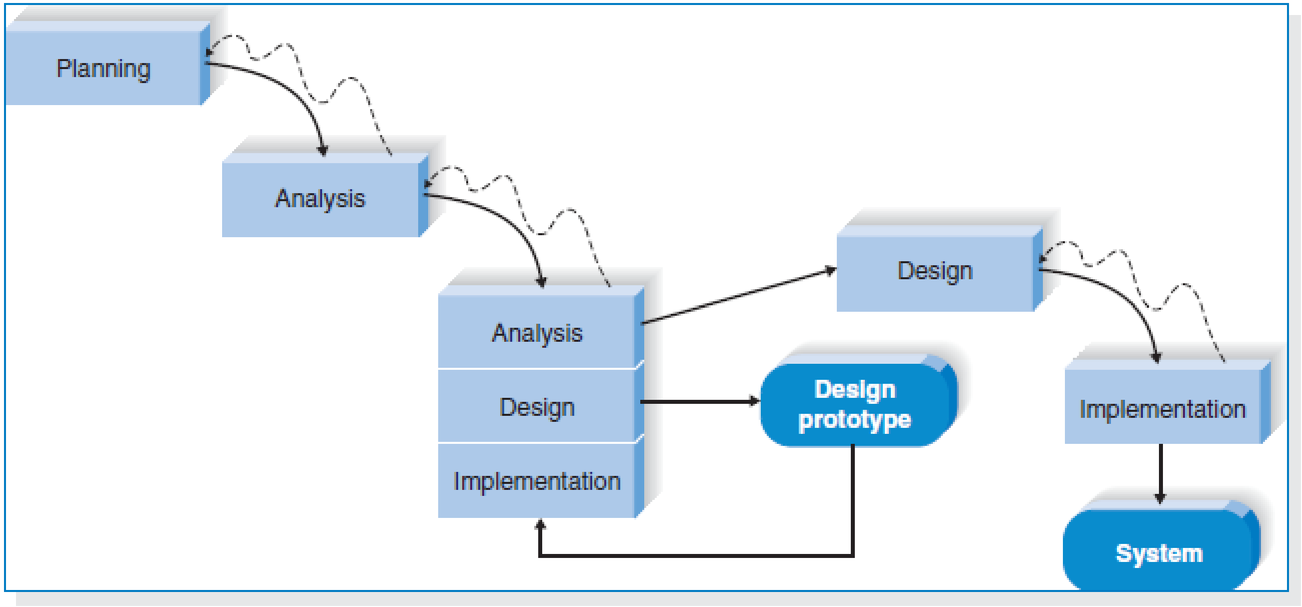


Throwaway prototyping

- Fairly thorough analysis for gather requirements.
- Developing ideas for the system concept.
- Many of the features suggested by the users may not be well understood.
- There may be challenging technical issues.
- Building design prototypes.

Throwaway prototyping

- It balances the benefits of well-thought-out analysis and design phases with the advantages of using prototypes to refine key issues before a system is built.



Agile development

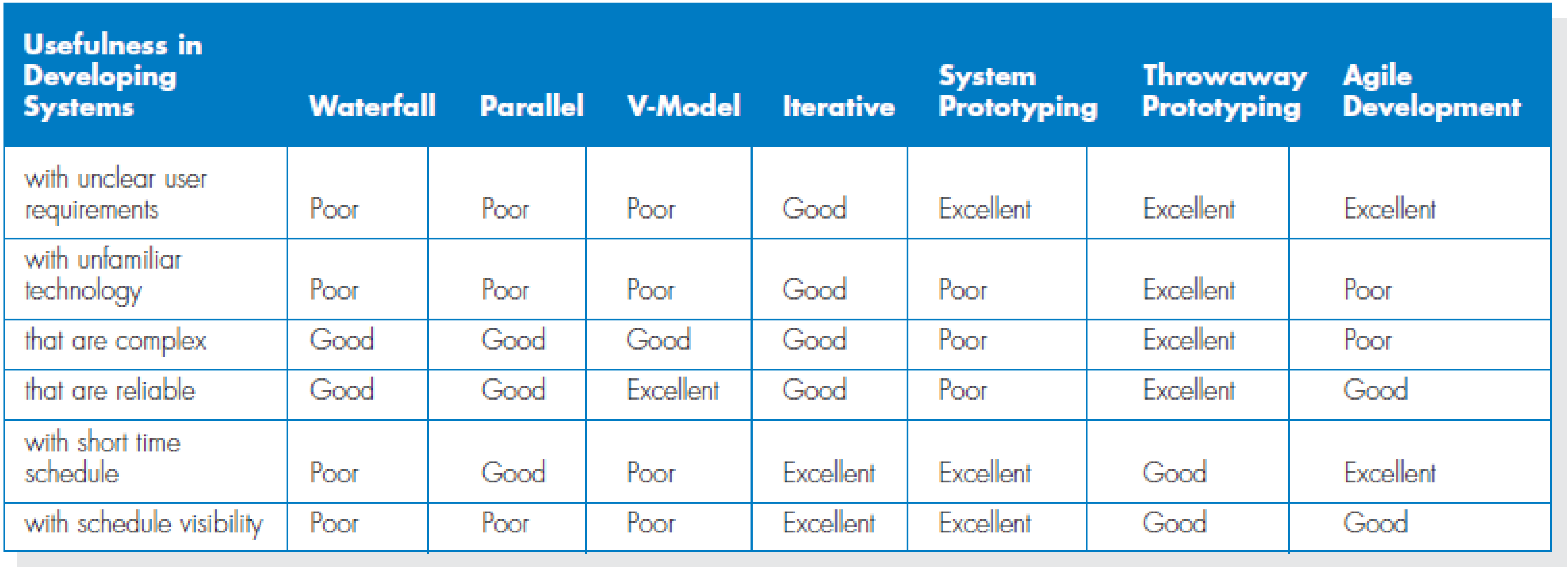
- A group of programming-centric methodologies.
- Focus on streamlining the SDLC.
- Much of the modelling and documentation overhead is eliminated.
- Face-to-face communication is preferred.
- Simple, iterative application development.
- Every iteration is a complete software project.
- There are several popular approaches to agile development

  Extreme programming, Scrum, and dynamic systems development method.

Extreme programming

- Emphasizes customer satisfaction and teamwork.
- Core values: communication, simplicity, feedback, and courage are.
- Designs are kept simple and clean.
- Early and frequent testing provides feedback.
- Response to changing requirements and technology.
- Begins with user stories that describe what the system needs to do.
- Then, programmers code in small, simple modules and test to meet those needs.

Agile vs. Waterfall



Usefulness in Developing Systems	Waterfall	Parallel	V-Model	Iterative	System Prototyping	Throwaway Prototyping	Agile Development
with unclear user requirements	Poor	Poor	Poor	Good	Excellent	Excellent	Excellent
with unfamiliar technology	Poor	Poor	Poor	Good	Poor	Excellent	Poor
that are complex	Good	Good	Good	Good	Poor	Excellent	Poor
that are reliable	Good	Good	Excellent	Good	Poor	Excellent	Good
with short time schedule	Poor	Good	Poor	Excellent	Excellent	Good	Excellent
with schedule visibility	Poor	Poor	Poor	Excellent	Excellent	Good	Good

Appropriate methodology

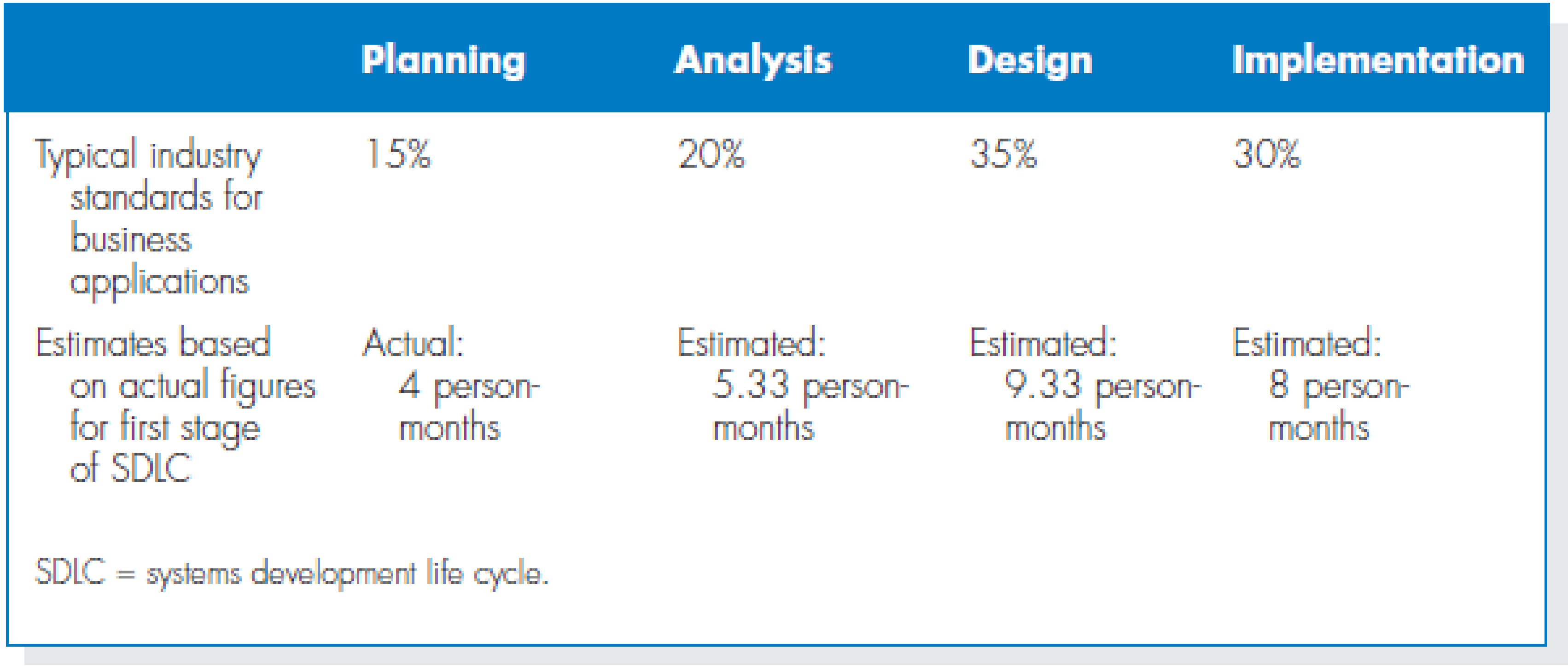
- Aspects:
  - Clarity of User Requirements;
  - Familiarity with Technology;
  - System Complexity;
  - System Complexity;
  - Short Time Schedules;
  - Schedule Visibility.

Project time frame

- Estimation can be performed manually or with the help of an estimation software.
- There are two basic ways:
  - Based on the time spent in the planning phase to predict the time required for the entire project;
  - The function point approach.

Project time frame

- Based on the time spent in the planning phase

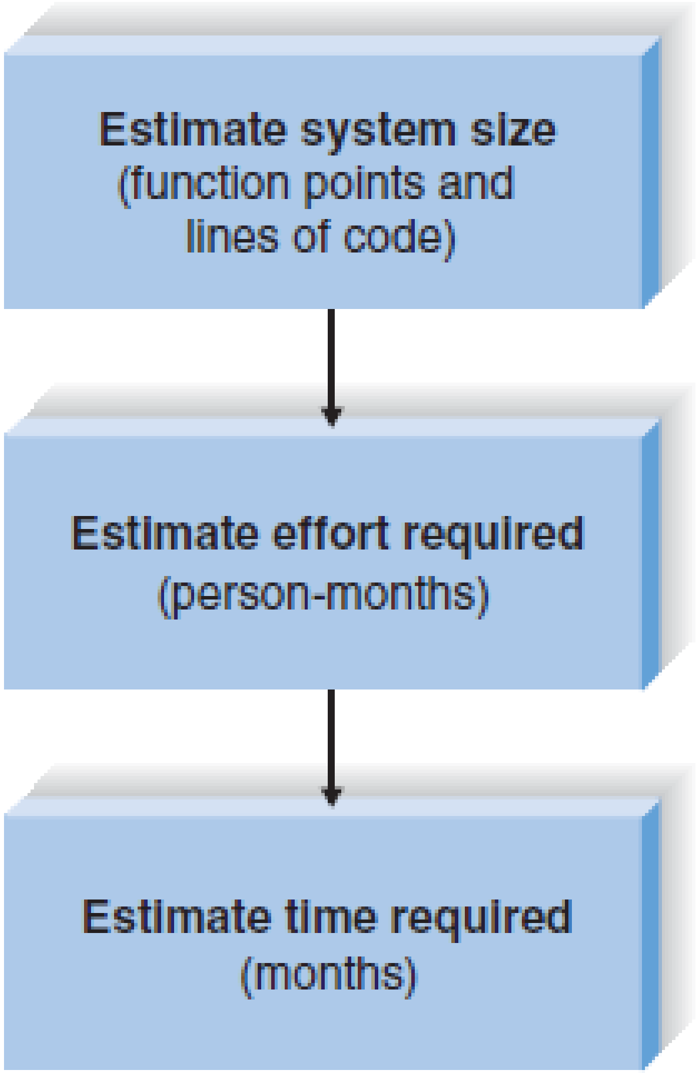


	Planning	Analysis	Design	Implementation
Typical industry standards for business applications	15%	20%	35%	30%
Estimates based on actual figures for first stage of SDLC	Actual: 4 person-months	Estimated: 5.33 person-months	Estimated: 9.33 person-months	Estimated: 8 person-months

SDLC = systems development life cycle.

Project time frame

- The function point approach.


Estimate system size (function points and lines of code)
Estimate effort required (person-months)
Estimate time required (months)

Developing the work plan

- Project manager:
  - Identifies the tasks that need to be accomplished;
  - Determines how long each one will take.
- The tasks are organized within a work breakdown structure.
- The work breakdown structure can be organized in one of two ways:
  - According to SDLC phases;
  - According to the different products to be developed.
- Project work plan:
  - Task information (task, deadline, status);
  - Dependencies;
  - Key milestones.

Staffing the project

- How many people.
- People's skills.
- Motivation.

- The more staff doesn't mean shorter project and keep team sizes under 8-10 people.
- Use appropriate report structure.
- Importance of technical and interpersonal skills.

Managing and controlling

- Refining estimates.
- Managing scope.
- Time boxing.
- Managing risk.

THANKS FOR ATTENTION!

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

EFOP-3.4.3-16-2016-00009

A felsőfokú oktatás minőségének és hozzáférhetőségének együttes javítása a Pannon Egyetemen

SYSTEM ANALYSIS AND SYSTEM DESIGN

3. REQUIREMENTS DETERMINATION

Imre Budai

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

Objectives

- Explain the analysis phase of the SDLC.
- Describe the content and purpose of the requirements definition statement.
- Classify requirements correctly as business, user, functional, or nonfunctional requirements.
- Employ the requirement elicitation techniques of interviews, JAD sessions, questionnaires, document analysis, and observation.

Objectives

- Define the role that each requirement elicitation technique plays in determining requirements.
- Describe several analysis strategies that can help the analyst discover requirements.

Introduction

- It involves expanding the vision described in the system request into a thorough, detailed understanding of exactly what the new system needs to do.
- It results in:
  - **Detailed requirements definition statement;**
  - Use cases;
  - Process models;
  - Data model.

Analysis phase

- Breaking a whole into its parts to understand the parts':
  - Nature;
  - Function;
  - Interrelationships.
- Results of the planning phase:
  - Business goals;
  - Project's scope;
  - Project feasibility;
  - Initial work plan.
- These are the key inputs into the analysis phase.

Analysis phase

- Steps of the analysis:
  - Understand the existing situation (the as-is system);
  - Identify improvements;
  - Define requirements for the new system (the to-be system).
- First step can be skipped or done in a limited manner.
- Analyst needs strong critical thinking skills - the ability to recognize strengths and weaknesses and recast an idea in an improved form.

Example

- User statement:
  - The new system should “eliminate inventory stock-outs.”
- Requirements:
  - On-hand inventory levels update twice per day;
  - Immediate out-of-stock notification reaching on hand item reorder point;
  - Recommended supplier with every out-of-stock notification;
  - Produce a supply purchase order and send if for approval;
  - Send approved supply purchase order to the supplier.

Requirements

- Transformation of the system requests:
  - From: high level statement of business requirements;
  - Into: a precise list of what the new system must do to provide the needed value to the business.

- Steps in requirement analysis:
  - Requirement determination.
  - Requirement elicitation techniques.
  - Requirement analysis strategies.

Requirement determination

- What is a requirement?
- A requirement is simply a statement of:
  - What the system must do;
  - What characteristics it needs to have.

- **Business requirements**: the business needs;
- **User requirements**: the users need to do;
- **Functional requirements**: the software should do;
- **Non-functional requirements**: characteristics the system should have;
- **System requirements**: how the system should be built.

Requirement determination

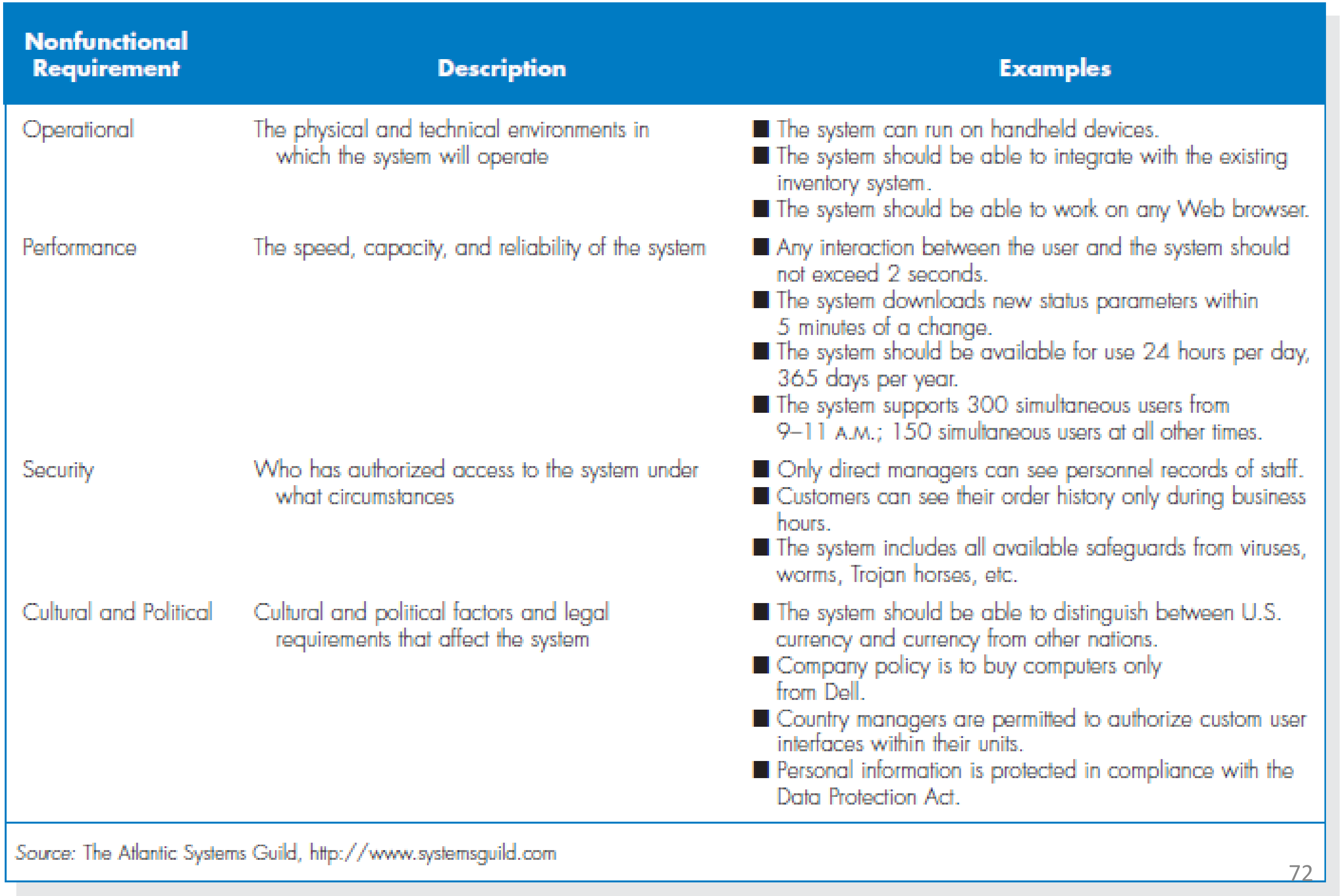
- Business requirements – example:
  - Shorten order processing time;
  - Reduce customer service costs.

- What the user actually needs to accomplish with the system – example:
  - Place a new customer order;
  - Re-order inventory.
- It leads to statements of the system’s functional requirements.

Requirement determination

- Non-functional requirements:
  - The quality attributes, design, and implementation constraints, and external interfaces which a product must have.
  - Important behavioural properties that the system must have, such as performance and usability.
- System characteristics:
  - Operational;
  - Performance;
  - Security;
  - Cultural;
  - Political.

Requirement determination



Nonfunctional Requirement	Description	Examples
Operational	The physical and technical environments in which the system will operate	■ The system can run on handheld devices. ■ The system should be able to integrate with the existing inventory system. ■ The system should be able to work on any Web browser.
Performance	The speed, capacity, and reliability of the system	■ Any interaction between the user and the system should not exceed 2 seconds. ■ The system downloads new status parameters within 5 minutes of a change. ■ The system should be available for use 24 hours per day, 365 days per year. ■ The system supports 300 simultaneous users from 9–11 A.M.; 150 simultaneous users at all other times.
Security	Who has authorized access to the system under what circumstances	■ Only direct managers can see personnel records of staff. ■ Customers can see their order history only during business hours. ■ The system includes all available safeguards from viruses, worms, Trojan horses, etc.
Cultural and Political	Cultural and political factors and legal requirements that affect the system	■ The system should be able to distinguish between U.S. currency and currency from other nations. ■ Company policy is to buy computers only from Dell. ■ Country managers are permitted to authorize custom user interfaces within their units. ■ Personal information is protected in compliance with the Data Protection Act.

Source: The Atlantic Systems Guild, http://www.systemsguild.com

Requirement determination

- Both, business and IT oriented users shall be involved in analysis phase:
  - Systems analysts may not understand the true business needs.
  - Business users may not be aware of the opportunities that a new technology may offer.
  - People who have an interest in the new system – often called stakeholders).

Requirement determination

- The first tasks for the analyst is to identify the primary sources of requirements, including:
  - The project sponsor;
  - Project champion(s);
  - All users of the system – both direct and indirect;
  - Possibly others.

- How best to elicit the requirements from the stakeholders?

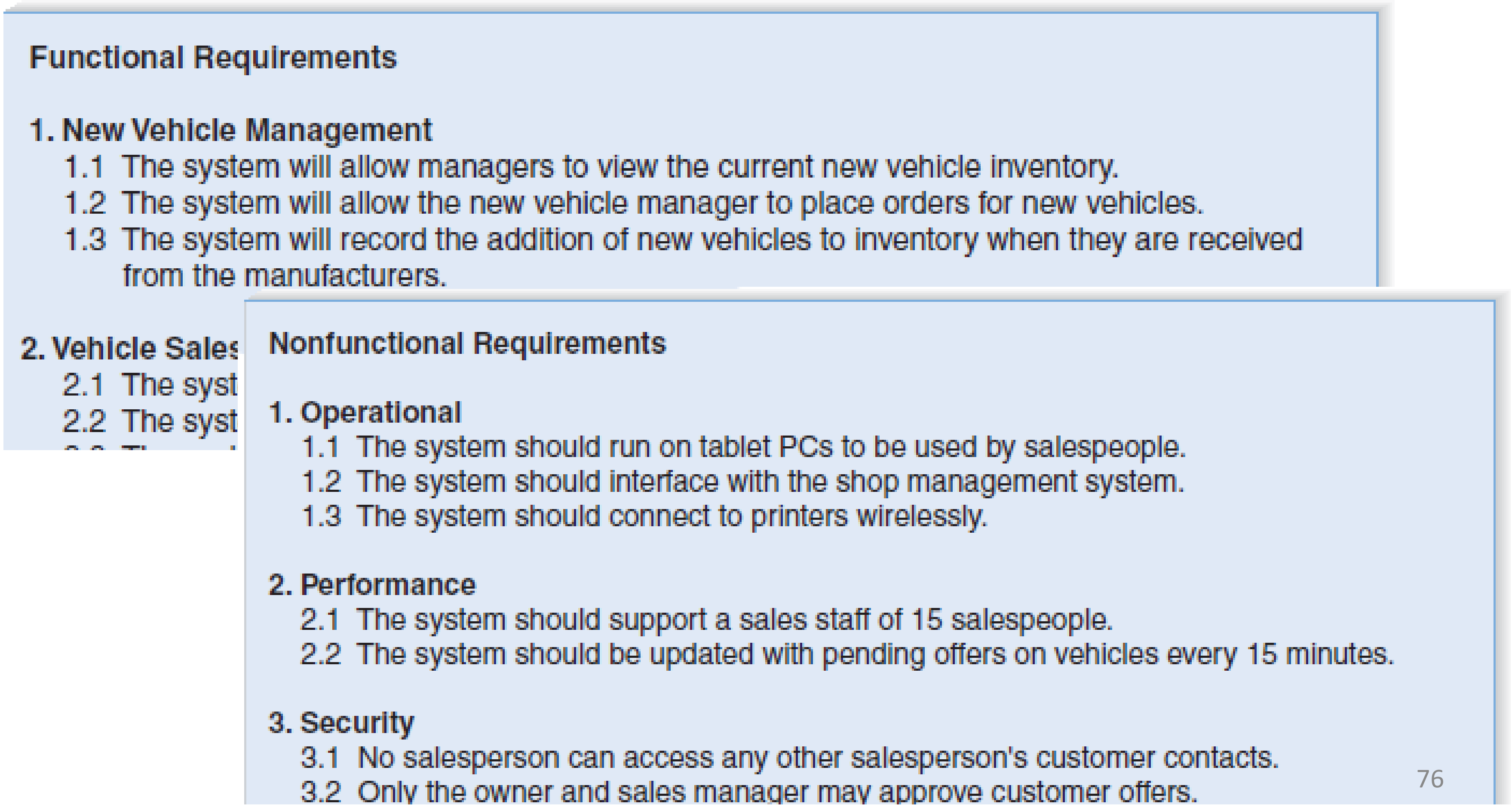
  Including interviews, questionnaires, observation, joint application development, and document analysis.

Requirement determination

- Analyst works with the entire project team and the business users to:
  - Verify;
  - Change;
  - Complete the list of requirements;
  - And prioritize the importance of the requirements, if necessary.
- The evolution of the requirements definition must be carefully managed.
- Keeping the requirements list tight and focused is a key to project success.

Requirement determination

- The requirements definition statement



Functional Requirements

1. New Vehicle Management

1.1 The system will allow managers to view the current new vehicle inventory.
1.2 The system will allow the new vehicle manager to place orders for new vehicles.
1.3 The system will record the addition of new vehicles to inventory when they are received from the manufacturers.

2. Vehicle Sales

2.1 The syst
2.2 The syst

Nonfunctional Requirements

1. Operational

1.1 The system should run on tablet PCs to be used by salespeople.
1.2 The system should interface with the shop management system.
1.3 The system should connect to printers wirelessly.

2. Performance

2.1 The system should support a sales staff of 15 salespeople.
2.2 The system should be updated with pending offers on vehicles every 15 minutes.

3. Security

3.1 No salesperson can access any other salesperson's customer contacts.
3.2 Only the owner and sales manager may approve customer offers.

Requirement elicitation

- Practical tips first:
  - Every contact and interaction between the analyst and a potential business user or manager is an opportunity to generate interest, enthusiasm, and commitment to the project.
  - The analyst should carefully determine who is included in the requirements definition process. All of the key stakeholders must be included.

Requirement elicitation

- Used techniques:
  - Interviews;
  - Joint Application Development (JAD);
  - Questionnaires;
  - Document analysis;
  - Observation.

Requirement elicitation

- Interviews:
  - Selecting interviewees and schedule;
  - Designing interview questions;
    - Closed-ended questions;
    - Open-ended questions;
    - Probing questions;
  - Top-down or Bottom-up interview;
  - Preparing for the interview;
  - Conducting the interview;
  - Post-interview follow-up;

Requirement elicitation

- Joint application development:
  - An information gathering technique – by IBM.
  - The facilitator is a person who:
    - Sets the meeting agenda;
    - Guides the discussion;
    - Does not join in the discussion as a participant.
  - The JAD group meets for several hours, several days, etc. until all of the issues have been discussed and the needed information is collected.
  - Not a very efficient way to collect information.

Requirement elicitation

- Joint application development:
  - Participant selection is similar to interviews.
  - JAD sessions can run from as little as a half day to several weeks.
  - Preparing JAD session.
  - Conducting the JAD Session.
    - The group sticks to the agenda.
    - The facilitator must help the group.
    - The facilitator records the group’s input.
    - The facilitator must remain neutral at all times.
  - Post-JAD follow-up

Requirement elicitation

- Questionnaires:
  - Selecting participants – sample or representative subset.
  - Designing the questionnaire.
  - Administering the questionnaire.
    - Clear explanation of the purpose and selection;
    - Due-date;
    - Offering to supply a summary of the questionnaire responses.
  - Questionnaire follow-up.

Requirement elicitation

- Document analysis:
  - Document analysis to understand the as-is system.
  - Most systems are not well documented.
  - Helpful documents can be:
    - paper reports, memorandums, policy manuals, user training manuals, organization charts, forms, etc.
    - Problem reports filed by the system users

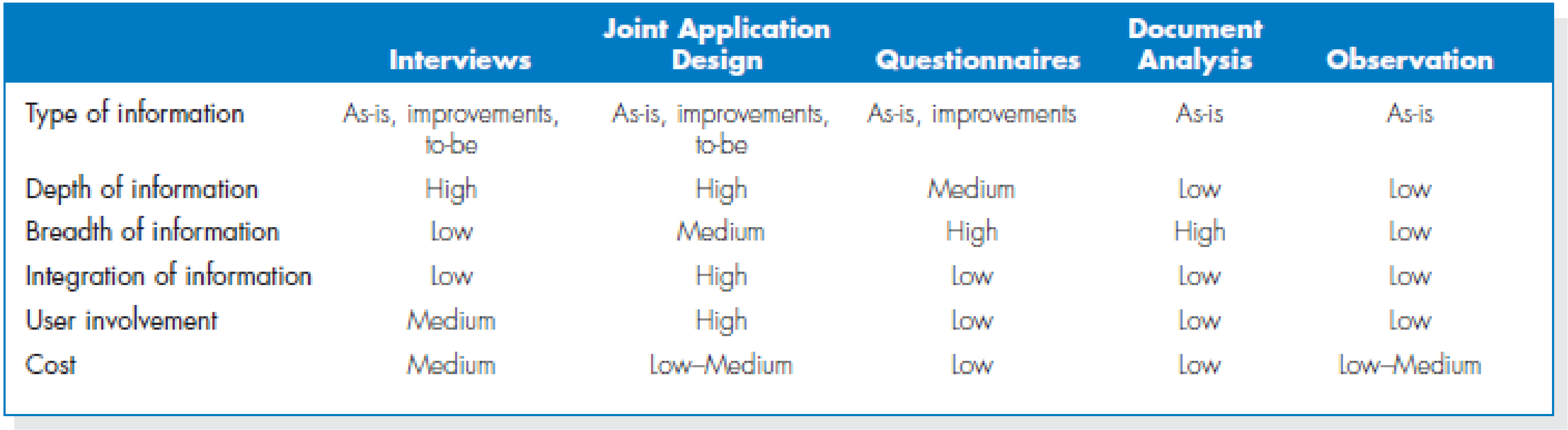
  - The most powerful indication that the system needs to be changed is when users create their own forms or add additional information to existing ones.

Requirement elicitation

- Observation:
  - Observation, the act of watching processes being performed.
  - It is a good way to check the validity of information gathered.

Requirement elicitation

- Comparison of elicitation techniques:



	Interviews	Joint Application Design	Questionnaires	Document Analysis	Observation
Type of information	As-is, improvements, to-be	As-is, improvements, to-be	As-is, improvements	As-is	As-is
Depth of information	High	High	Medium	Low	Low
Breadth of information	Low	Medium	High	High	Low
Integration of information	Low	High	Low	Low	Low
User involvement	Medium	High	Low	Low	Low
Cost	Medium	Low–Medium	Low	Low	Low–Medium

Requirement analysis strategies

- Problem analysis.
- Root cause analysis.
- Duration analysis.
- Activity-based costing (ABC).
- Informal benchmarking.
- Outcome analysis.
- Technology analysis.
- Activity elimination

EFOP-3.4.3-16-2016-00009

A felsőfokú oktatás minőségének és hozzáférhetőségének együttes javítása a Pannon Egyetemen

THANKS FOR ATTENTION!

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

EFOP-3.4.3-16-2016-00009

A felsőfokú oktatás minőségének és hozzáférhetőségének együttes javítása a Pannon Egyetemen

SYSTEM ANALYSIS AND SYSTEM DESIGN

4. USE CASE ANALYSIS

Imre Budai

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

Objectives

- Explain the purpose of use cases in the analysis phase of the SDLC.
- Describe the various parts of a use case and the purpose of each part.
- Explain the process used to create a use case.
- Describe how use cases contribute to the functional requirements.
- Describe how use cases inform the development of test plans.

Introduction

- System proposal finished in the requirements determination.

  It defines exactly what the new system should do.
- In this chapter, we discuss use cases as a means of expressing user requirements.
- What the users intend to do with the system.
- It helps to understand and clarify the users’ required interactions.

Introduction

- Difficulties:
  - Users may not know what is (not) possible for the system to do.
  - Users may have difficulty envisioning new ways to redesign business processes - because they are so accustomed to things being done the “old way.”
  - It is common for users to describe things they think they want from the new system, but our focus should be on the real needs for the new system.
  - Users often found it difficult to learn the process and data modelling languages used by the analysts.

Use cases

- It depicts a set of activities performed to produce some output result.
- It describes:
  - How an external user triggers an event;
  - All events trigger a set of activities the system must perform.
- This event-driven modelling can be thought of as a response to some trigger event.

Use cases

- It is important to create use cases:
  - Reengineering processes;
  - Making any (significantly altering) changes to business processes.
- This event-driven modelling can be thought of as a response to some trigger event.
- It describes what the system will do from the user’s perspective.

Elements

- Elements of a use case:
  - Basic information:
    - Name;
    - Number;
    - Description;
    - Priority;
    - Actor – or user role;
    - Trigger.

Elements

- Elements of a use case:
  - Preconditions:
    - Use cases are often performed in a sequence;
    - Clearly define what needs to be accomplished before each use case begins - state the system must be in before;
  - Normal course:
    - The major steps that are performed;
    - The inputs used for the steps;
    - The outputs produced by the steps;
    - The normal course lists the steps that are performed when everything flows smoothly in the system.

Elements

- Elements of a use case:
  - Alternative courses:
    - Alternative courses also will lead to a successful conclusion of the use case;
    - The location where it differs in logic from the normal course is clearly stated;
    - It also depicts two potential paths through these steps;
  - Post conditions:
    - It defines the final products of this use case;
    - These post conditions also serve to define the preconditions for the next use case in the series.

Elements

- Elements of a use case:
  - Exceptions:
    - A use case should describe any error conditions or exceptions that may occur;
    - These are not normal branches in decision logic;
    - but are unusual occurrences or errors;
    - That could potentially be encountered;
    - And will lead to an unsuccessful result.
    - We want to be sure that the system does not fail while in use because of an error that no one thought about.

Elements

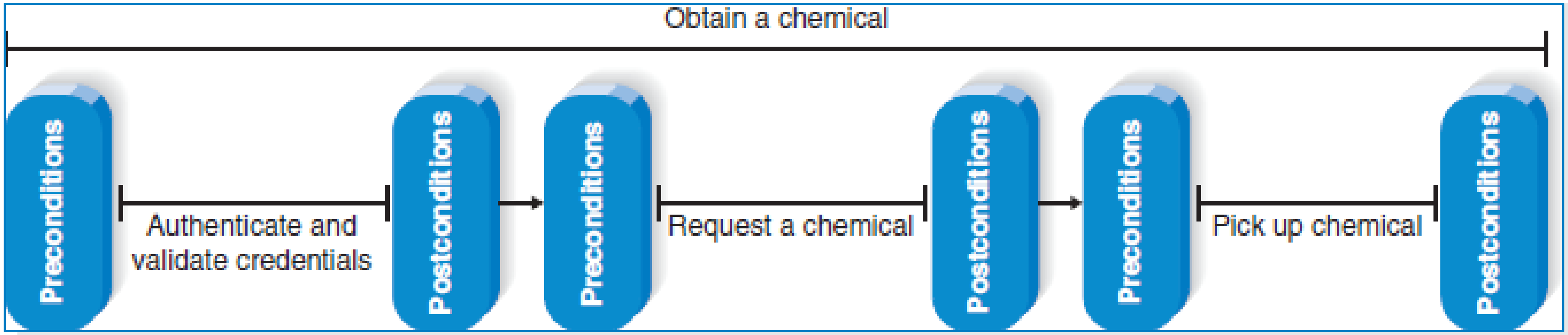
- Elements of a use case:
  - Summary inputs and outputs:
    - It summarizes the set of major inputs and outputs to the steps of the use case;
    - Major inputs and outputs;
    - Along with its source or destination;
    - These are all possible inputs and outputs, not just those that are part of the normal course.

Elements

- Elements of a use case:
  - Additional issues:
    - If appropriate, it may be helpful to include sections devoted to:
      - Frequency of use;
      - Business rules;
      - Special requirements;
      - Assumptions;
      - Notes and issues;

Use cases

- It is important to know exactly what state the system should be in before and after the use case
  - Different sources before
  - Multiple paths after
- That is the purpose of the precondition and post condition sections of the use case.



Systems Analysis Design 5th edition p.154.

Functional requirements

- Use cases do explain the user’s interaction with the system.
- Use cases only convey the user’s point of view.
- They omit a lot of details that are necessary to know before the system can be developed.
- Systems analysts’ role: Transforming the user’s view into the developer’s view by creating functional requirements.

Use cases and testing

- Developing test plans early in the development process.
  - By studying the use cases;
  - And the functional requirements derived from them.
- Elements of the tests can be identified those will be performed when the system enters testing.
- Well prepared testing personnel.
- Not forced to develop and perform the tests in a rush.
- Helpful suggestions by quality assurance personnel.
- Valuable feedback early in the development process.

Building a use case

1. Identify the use cases

   Start a use case report form for each use case by filling in the name, description and trigger.

2. Identify the major steps within each use case

   For each use case, fill in the major steps needed to complete the task.

3. Identify elements within steps

   For each steps, identify its triggers and its inputs.

4. Confirm the use case

   For each use case, validate that it is correct and complete.

Identify the use cases

- Ask who, what, when, and where about the use cases (or tasks).
  - What are the major tasks that are performed?
  - What triggers this task?
  - What tells you to perform this task?

Identify major steps

- Ask how about each use case.
  - What information/forms/reports do you need to perform this task?
  - Who gives you these information/forms/reports?
  - What information/forms/report does this produce and where do they go?
  - How do you produce this report?
  - How do you change the information on the report?
  - How do you process forms?
  - What tools do you use to do this step (e.g., paper, e-mail, phone)?

Identify elements

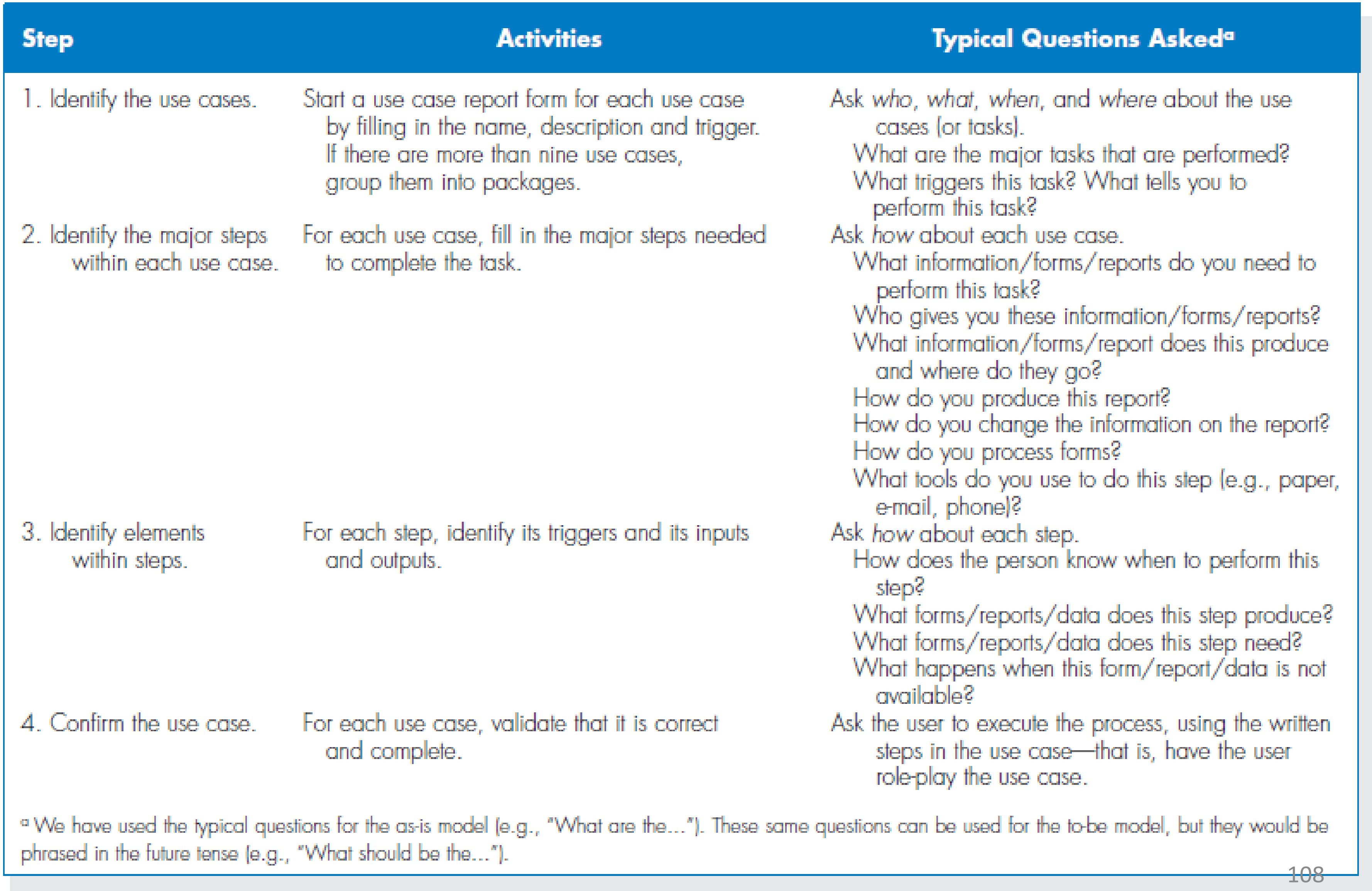
- Ask how about each step.
  - How does the person know when to perform this step?
  - What forms/reports/data does this step produce?
  - What forms/reports/data does this step need?
  - What happens when this form/report/data is not available?

Confirming the use case

- Ask the user to execute the process, using the written steps in the use case — that is, have the user role-play the use case.

- Revise functional requirements based on use cases.
- The functional requirements in the requirements definition may be modified:
  - To reflect this more detailed understanding;
  - To provide insight to the development team on some “back-end” processing;

Building a use case



Step	Activities	Typical Questions Asked[a]
1. Identify the use cases.	Start a use case report form for each use case by filling in the name, description and trigger. If there are more than nine use cases, group them into packages.	Ask *who*, *what*, *when*, and *where* about the use cases (or tasks). What are the major tasks that are performed? What triggers this task? What tells you to perform this task?
2. Identify the major steps within each use case.	For each use case, fill in the major steps needed to complete the task.	Ask *how* about each use case. What information/forms/reports do you need to perform this task? Who gives you these information/forms/reports? What information/forms/report does this produce and where do they go? How do you produce this report? How do you change the information on the report? How do you process forms? What tools do you use to do this step (e.g., paper, e-mail, phone)?
3. Identify elements within steps.	For each step, identify its triggers and its inputs and outputs.	Ask *how* about each step. How does the person know when to perform this step? What forms/reports/data does this step produce? What forms/reports/data does this step need? What happens when this form/report/data is not available?
4. Confirm the use case.	For each use case, validate that it is correct and complete.	Ask the user to execute the process, using the written steps in the use case—that is, have the user role-play the use case.

[a] We have used the typical questions for the as-is model (e.g., "What are the..."). These same questions can be used for the to-be model, but they would be phrased in the future tense (e.g., "What should be the...").

Developed use cases

- Developing test plans early in the development process.

THANKS FOR ATTENTION!

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

EFOP-3.4.3-16-2016-00009

A felsőfokú oktatás minőségének és hozzáférhetőségének együttes javítása a Pannon Egyetemen

SYSTEM ANALYSIS AND SYSTEM DESIGN

5. PROCESS MODELLING

Imre Budai

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

Objectives

- Explain the rules and style guidelines for data flow diagrams.
- Describe the process used to create data flow diagrams.
- Create data flow diagrams.
- Using BPMN

Process model

- A process model is a graphical way of representing how a business system should operate.
- It illustrates the processes or activities that are performed and how data move among them.
- A process model can be used to
  - Document the current system (i.e., as-is system) or
  - The new system being developed (i.e., to-be system), whether computerized or not.

Data flow diagramming

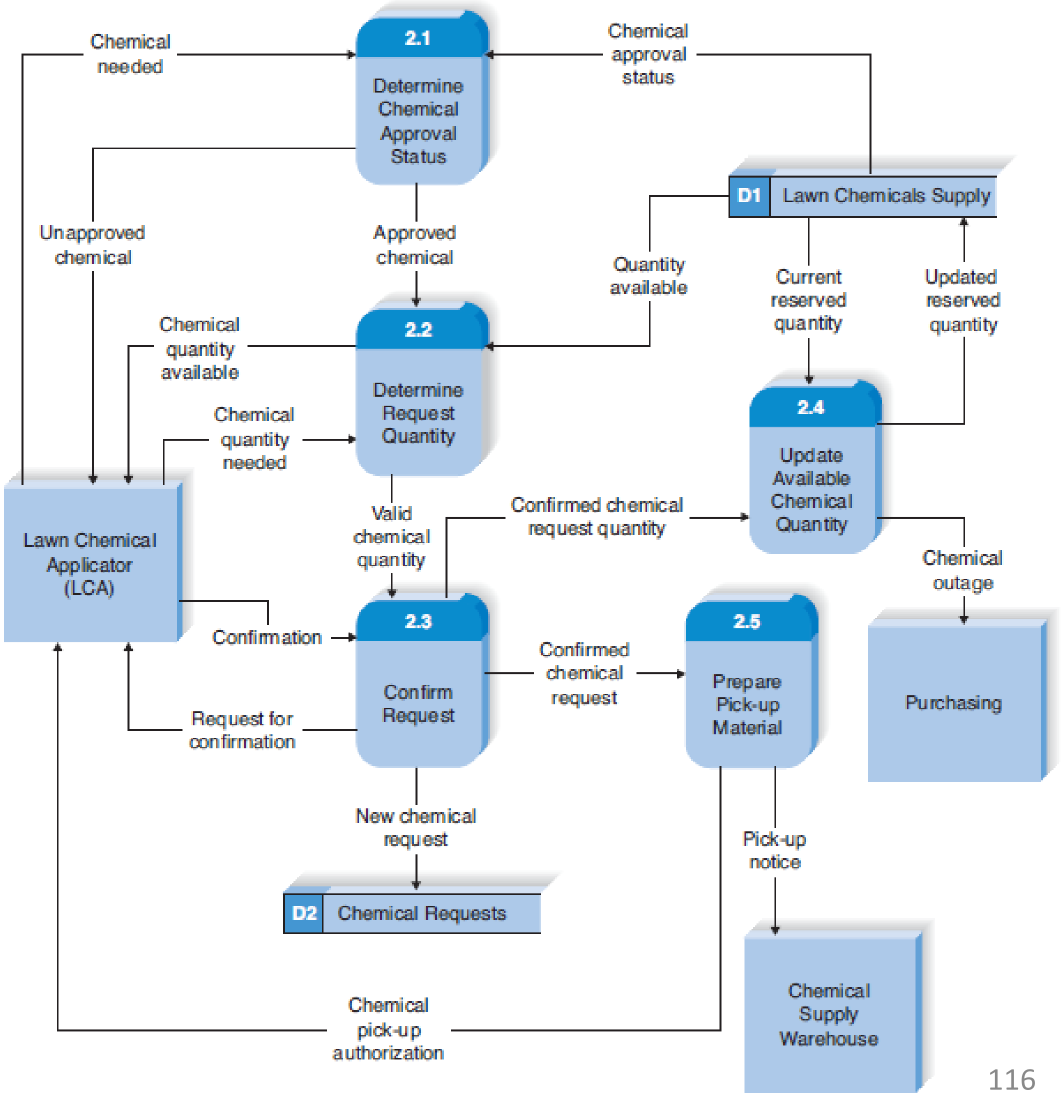
- One of the most commonly used technique
- Data flow diagramming is a technique that diagrams the business processes and the data that pass among them.
- Although the name data flow diagram (DFD) implies a focus on data, this is not the case.
    - The focus is mainly on the processes or activities that are performed.

BPMN

- It is a graphical notation that depicts the steps in a business process.
- BPMN depicts the end to end flow of a business process.
- The notation has been specifically designed to coordinate the sequence of processes and the messages that flow between different process participants in a related set of activities.
- It can be used:
  - For a high level for business users
  - At a lower level for process implementers.

DFD

- An example



DFD

- There are four symbols in the DFD language
  - Processes
  - Data flows
  - Data stores
  - External entities
- Each of them is represented by a different graphic symbol.

Process

- A process is an activity or a function that is performed for some specific business reason.
- Processes can be manual or computerized.
- Every process should be named starting with a verb and ending with a noun (e.g., “Determine request quantity”).
- Each process performs only one activity.
- Every process must have at least
  - One input data flow and
  - One output data flow.

Data flow

- A data flow is a single piece of data (e.g., quantity available) or a logical collection of several pieces of information (e.g., new chemical request).
- Every data flow should be named with a noun.
- The description of a data flow lists exactly what data elements the flow contains.
- Data flows are the glue that holds the processes together.
  - One end of every data flow will always come from or go to a process,
  - With the arrow showing the direction into or out of the process.
- It shows what inputs go into each process and what outputs each process produces.

Data store

- A data store is a collection of data that is stored in some way.
- Every data store is named with a noun and is assigned an identification number and a description.
- It forms the starting point for the data model.
- It is the principal link between the process model and the data model.
    - Data flows coming out of a data store indicate that information is retrieved from the data store.
    - Data flows going into a data store indicate that information is added to the data store.
    - Data flows going both into and out of a data store indicate that information in the data store is changed.

Data store

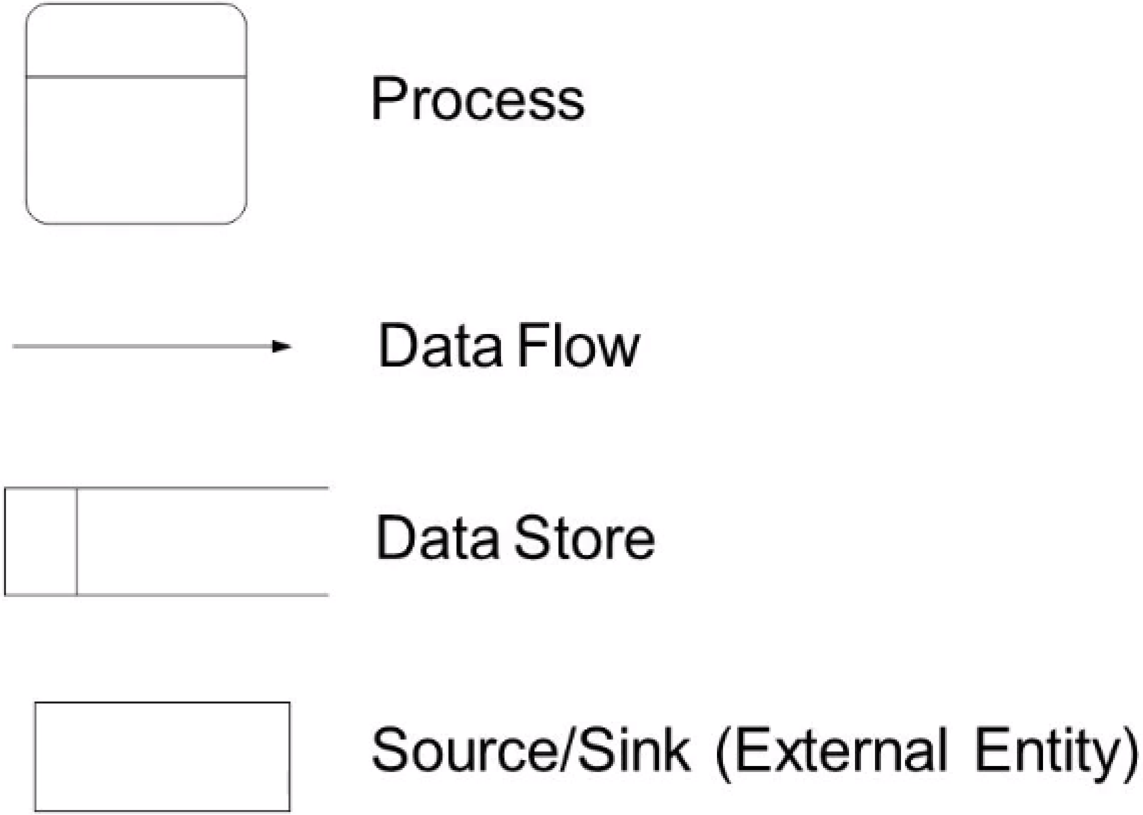
- All data stores must have at least one input data flow, unless they are created and maintained by another information system or on another page of the DFD.
- All data stores have at least one output data flow on some page of the DFD.

External entity

- An external entity interacts with the system, however it is external to it. It can be a:
  - Person
  - Organization, organization unit
  - System.
- It typically corresponds to the primary actor identified in the use case.
- External entities provide data to the system or receive data from the system, and serve to establish the system boundaries.
- May or may not be part of the organization.

Elements

- Graphical representation:



Process

Data Flow

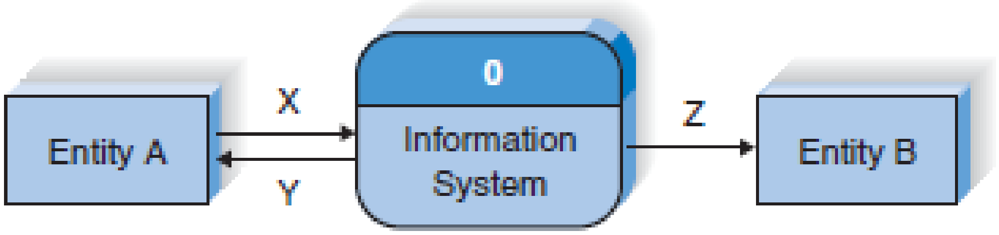
Data Store

Source/Sink (External Entity)

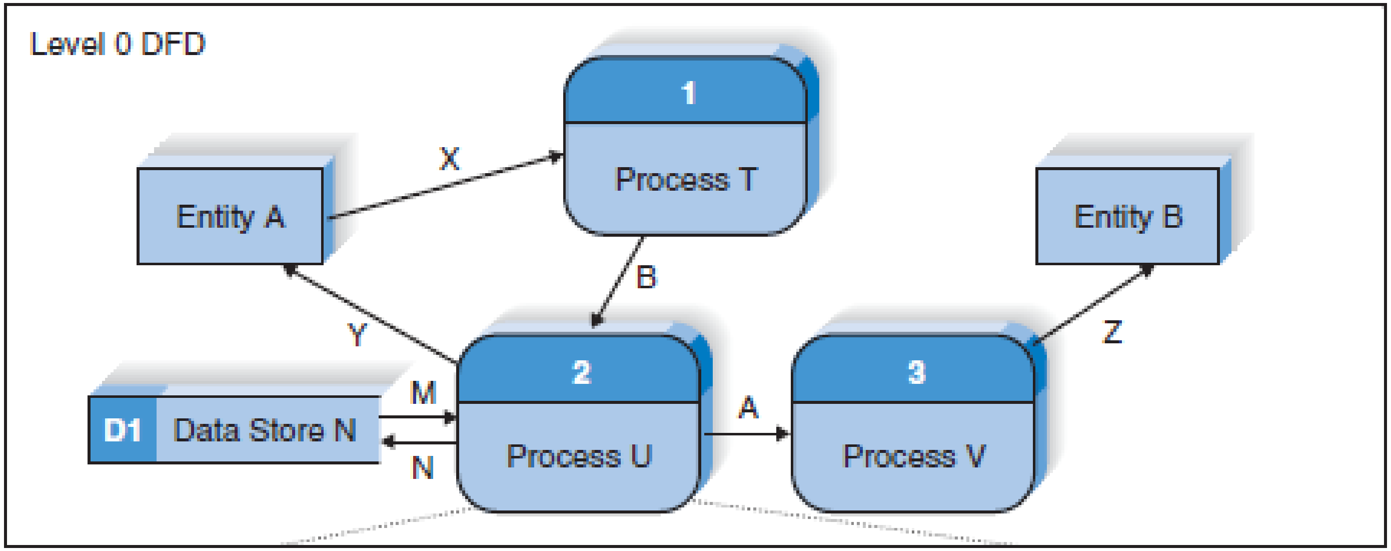
Decomposition

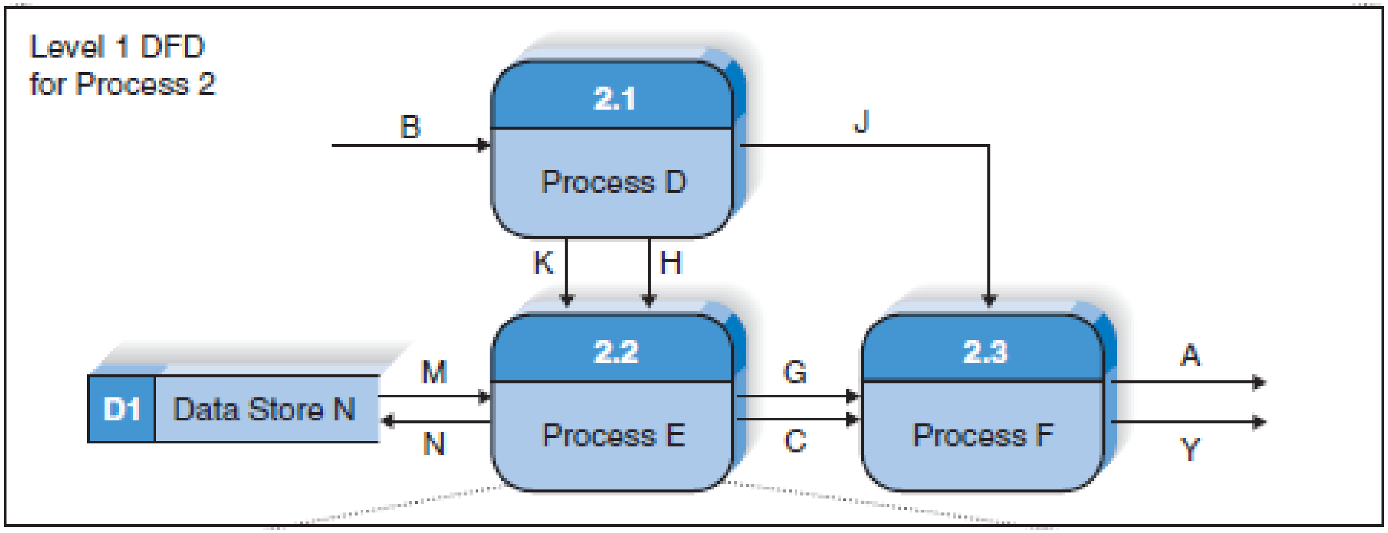
- Most business processes are too complex to be explained in one DFD.
- Therefore they are composed of a set of DFDs.
- The first DFD provides a summary of the overall system, with additional DFDs providing more and more detail about each part of the overall business process.

- Context diagram:



Decomposition

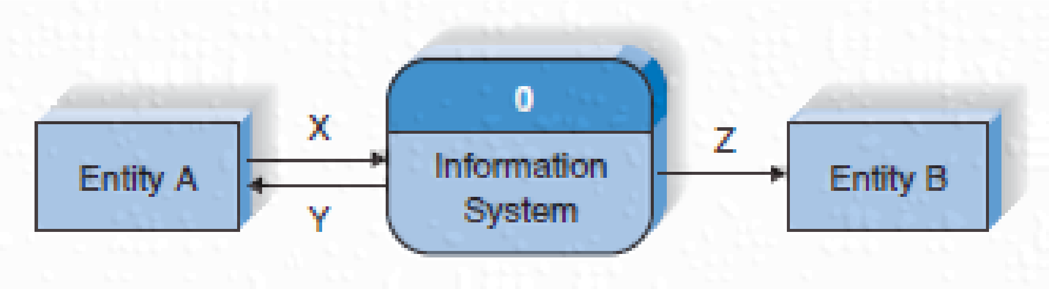




Context diagram

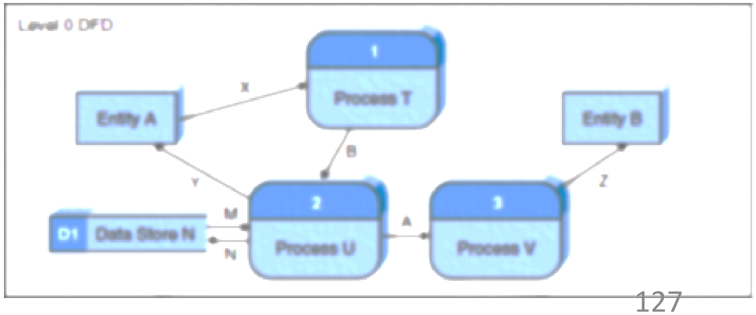
- Most business processes are too complex to be explained in one DFD.
- Therefore they are composed of a set of DFDs.
- The first DFD provides a summary of the overall system, with additional DFDs providing more and more detail about each part of the overall business process.

- Context diagram:



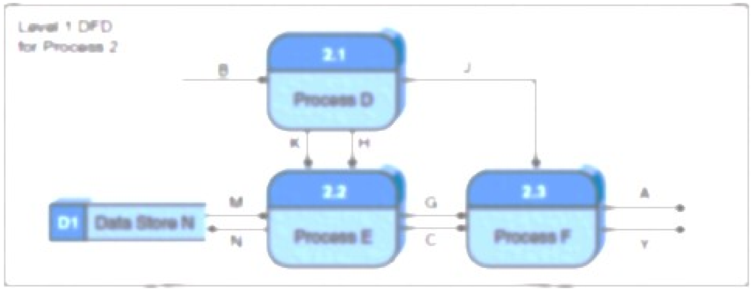
Level0 diagram

- Shows all the processes at the first level of numbering
  - Data stores
  - External entities
  - Data flows among them
- Purpose: show all the major high-level processes of the system and how they are interrelated.
- All process models have one and only one level 0 DFD.



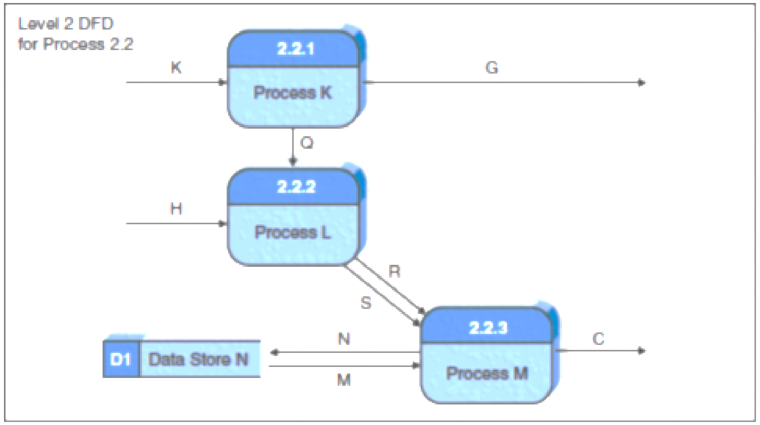
Level1 diagram

- Level 0 DFD shows only how the major high-level processes in the system interact.
- Each process on the level 0 DFD can be decomposed into a more explicit DFD → level 1 DFD
- It shows how it operates in greater detail.



Level2 diagram

- Lowest level of decomposition.
- It is sometimes difficult to remember which DFD level is which.
- It may help to remember that the level numbers refer to the number of decimal points in the process numbers on the DFD.
  - A level 0 DFD has process numbers with no decimal points.
  - level 2 DFD has numbers with two decimal points.



Creating DFD

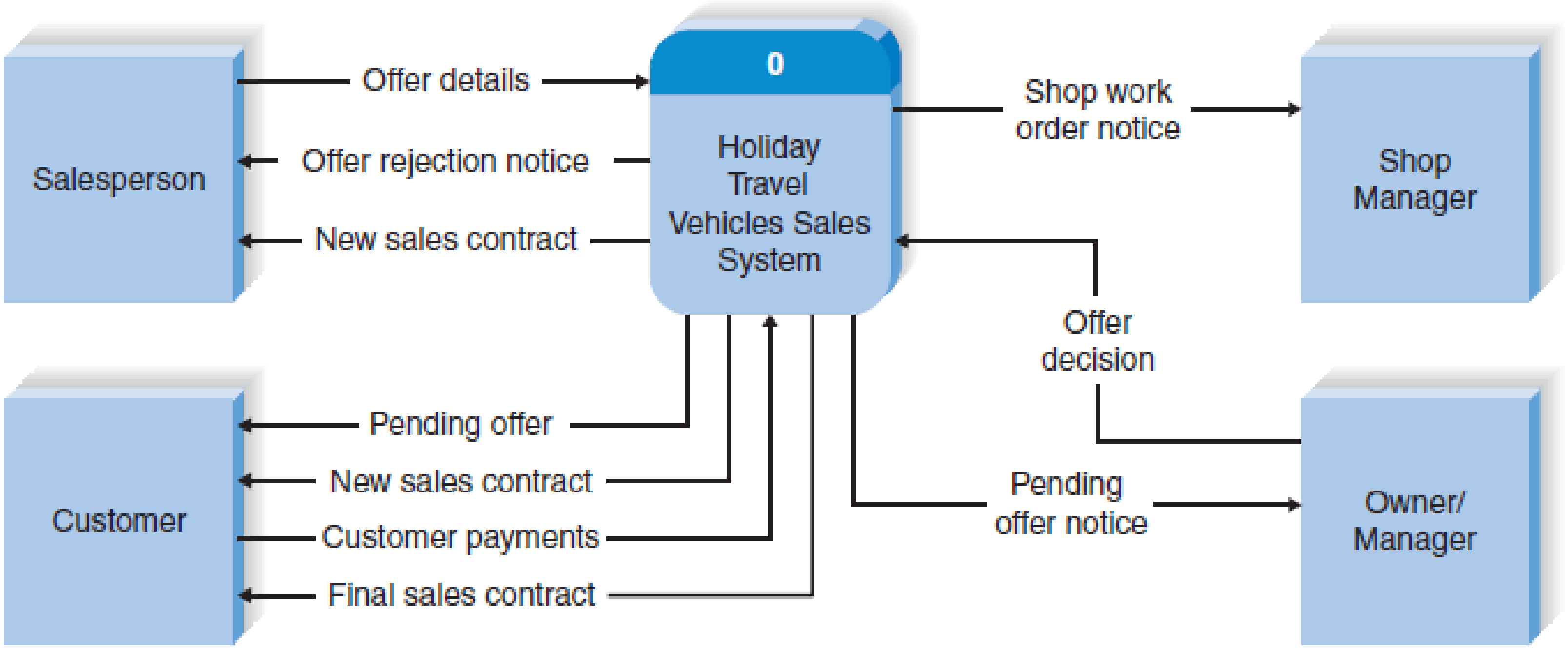
- Data flow diagrams start with the information in the use cases and the requirements definition.
- It is useful to first build the context diagram showing all the external entities and the data flows that originate from or terminate in them.
- Then creating DFD fragment for each use case that shows how the use case exchanges data flows with the external entities and data stores.
- These DFD fragments are organized into a level 0 DFD.

Creating DFD

- Developing level 1 DFDs, based on the steps within each use case, to better explain how they operate.
- In some cases, these level 1 DFDs are further decomposed into level 2 DFDs, level 3 DFDs, level 4 DFDs, and so on…
- Finally, validating the set of DFDs to make sure that they are complete and correct.

Creating DFD

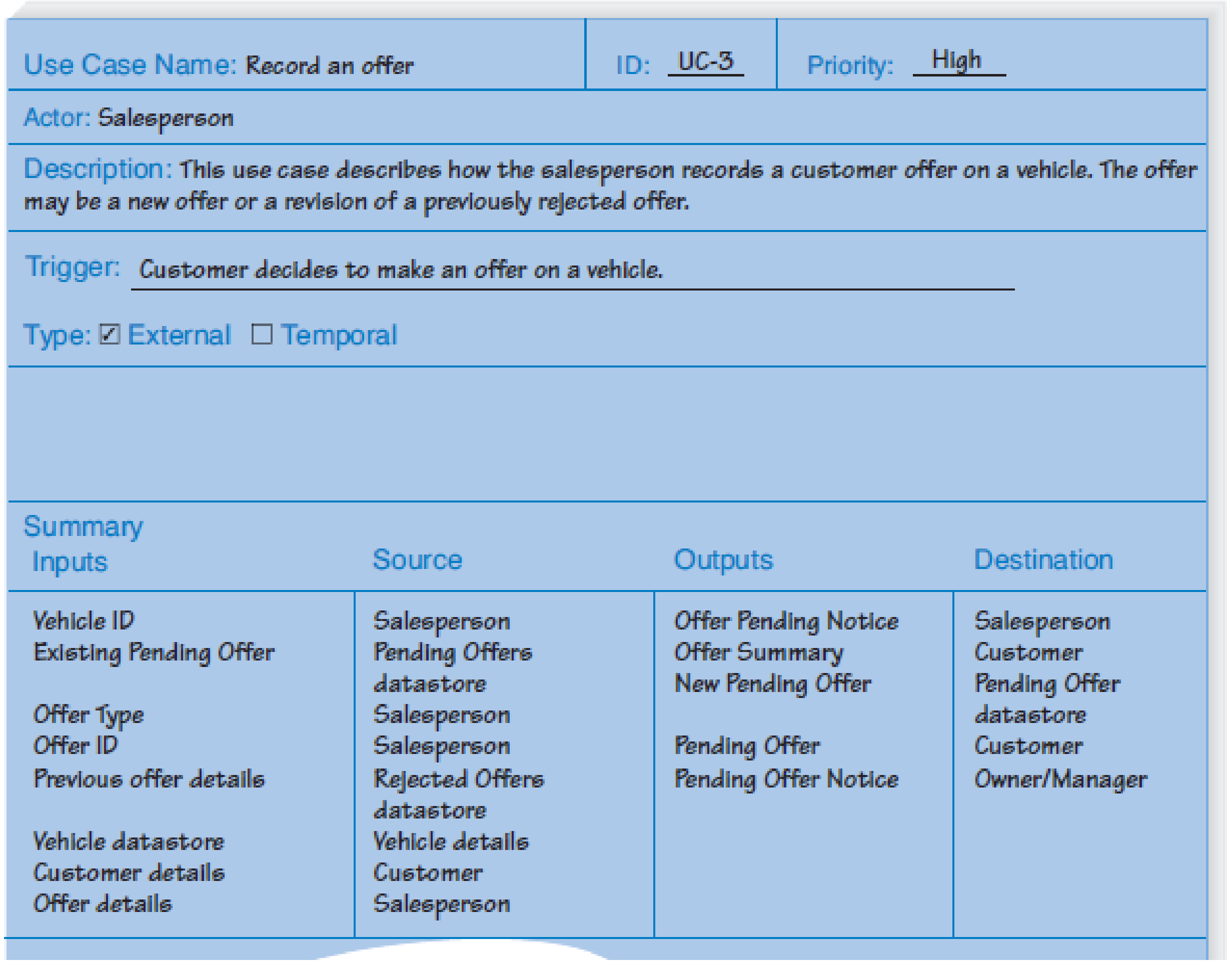
- Context diagram



Creating DFD

- Defragmenting

  Use Cases



Use Case Name: Record an offer | ID: UC-3 | Priority: High

Actor: Salesperson

Description: This use case describes how the salesperson records a customer offer on a vehicle. The offer may be a new offer or a revision of a previously rejected offer.

Trigger: Customer decides to make an offer on a vehicle.

Type: ☑ External ☐ Temporal

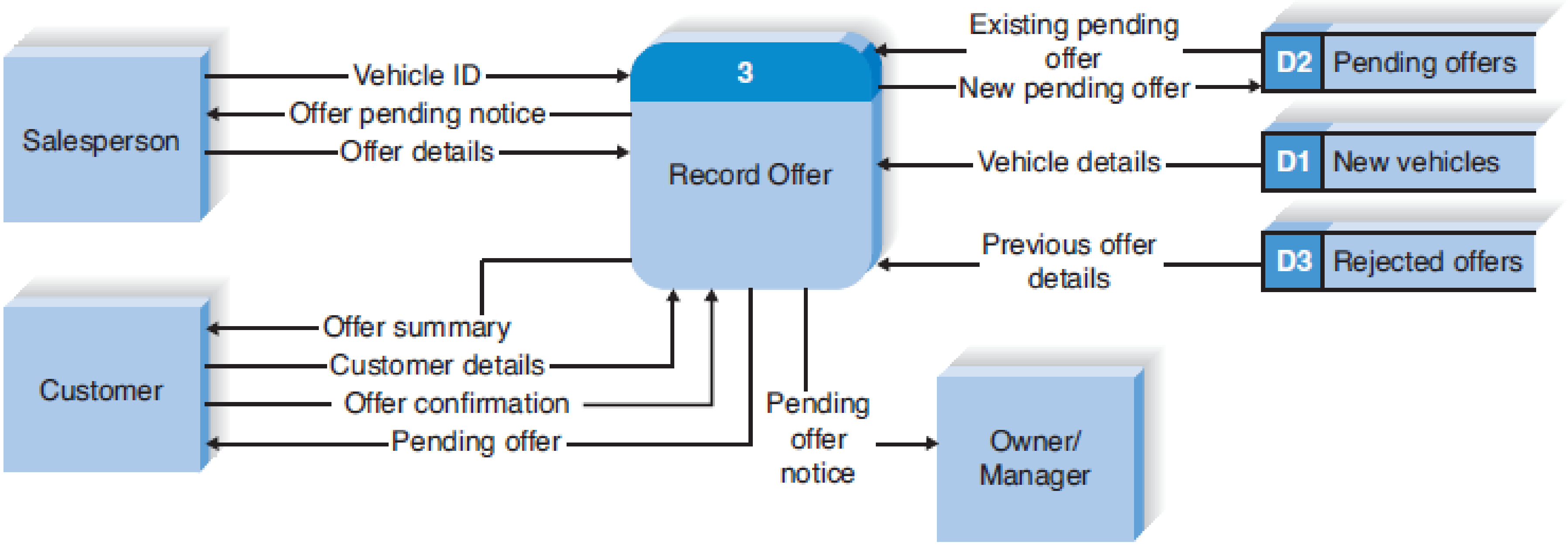
Summary

Inputs	Source	Outputs	Destination
Vehicle ID	Salesperson	Offer Pending Notice	Salesperson
Existing Pending Offer	Pending Offers datastore	Offer Summary	Customer
		New Pending Offer	Pending Offer datastore
Offer Type	Salesperson		
Offer ID	Salesperson	Pending Offer	Customer
Previous offer details	Rejected Offers datastore	Pending Offer Notice	Owner/Manager
Vehicle datastore	Vehicle details		
Customer details	Customer		
Offer details	Salesperson		

Creating DFD

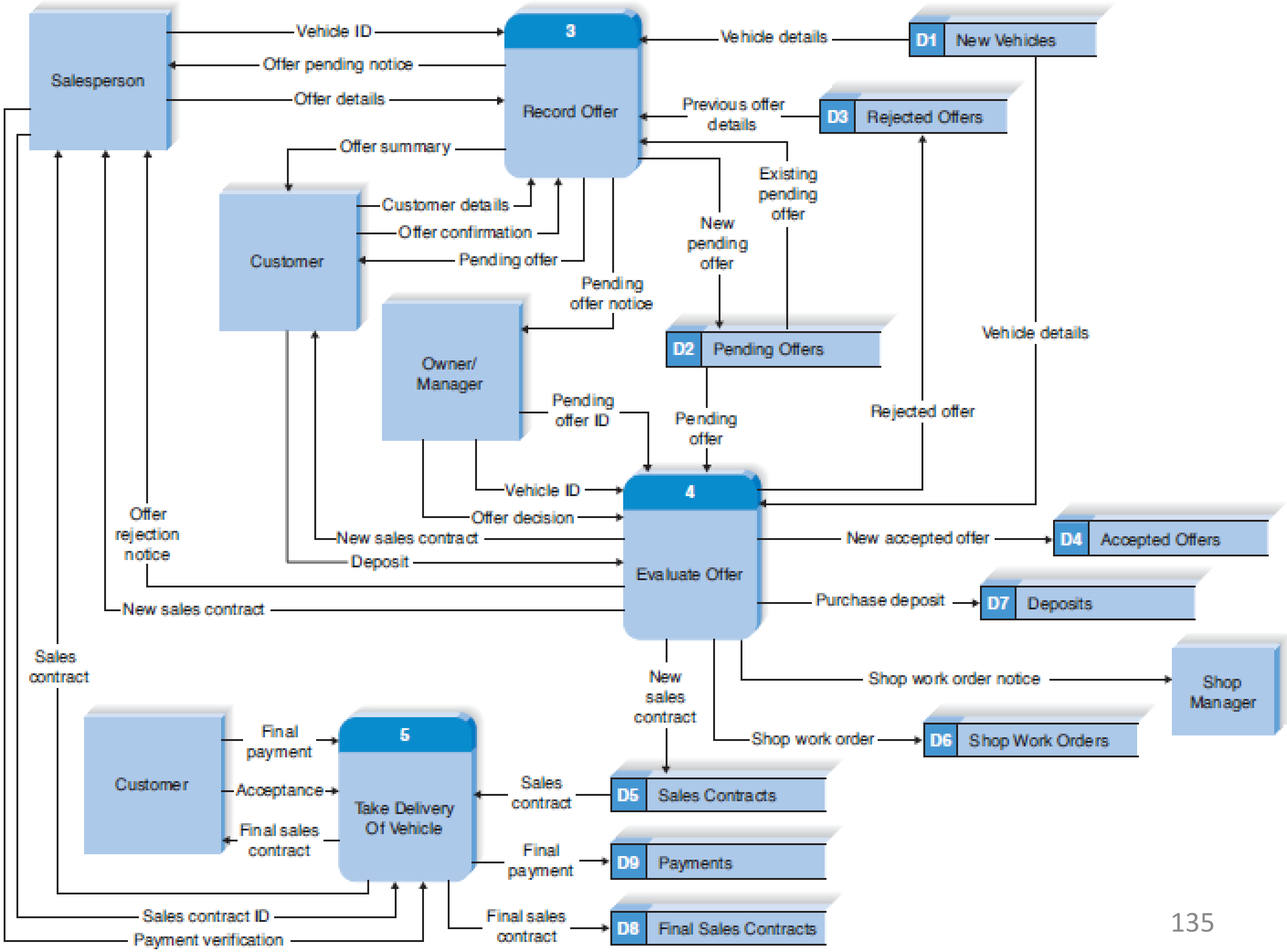
EFOP-3.4.3-16-2016-00009

A felsőfokú oktatás minőségének és hozzáférhetőségének együttes javítása a Pannon Egyetemen



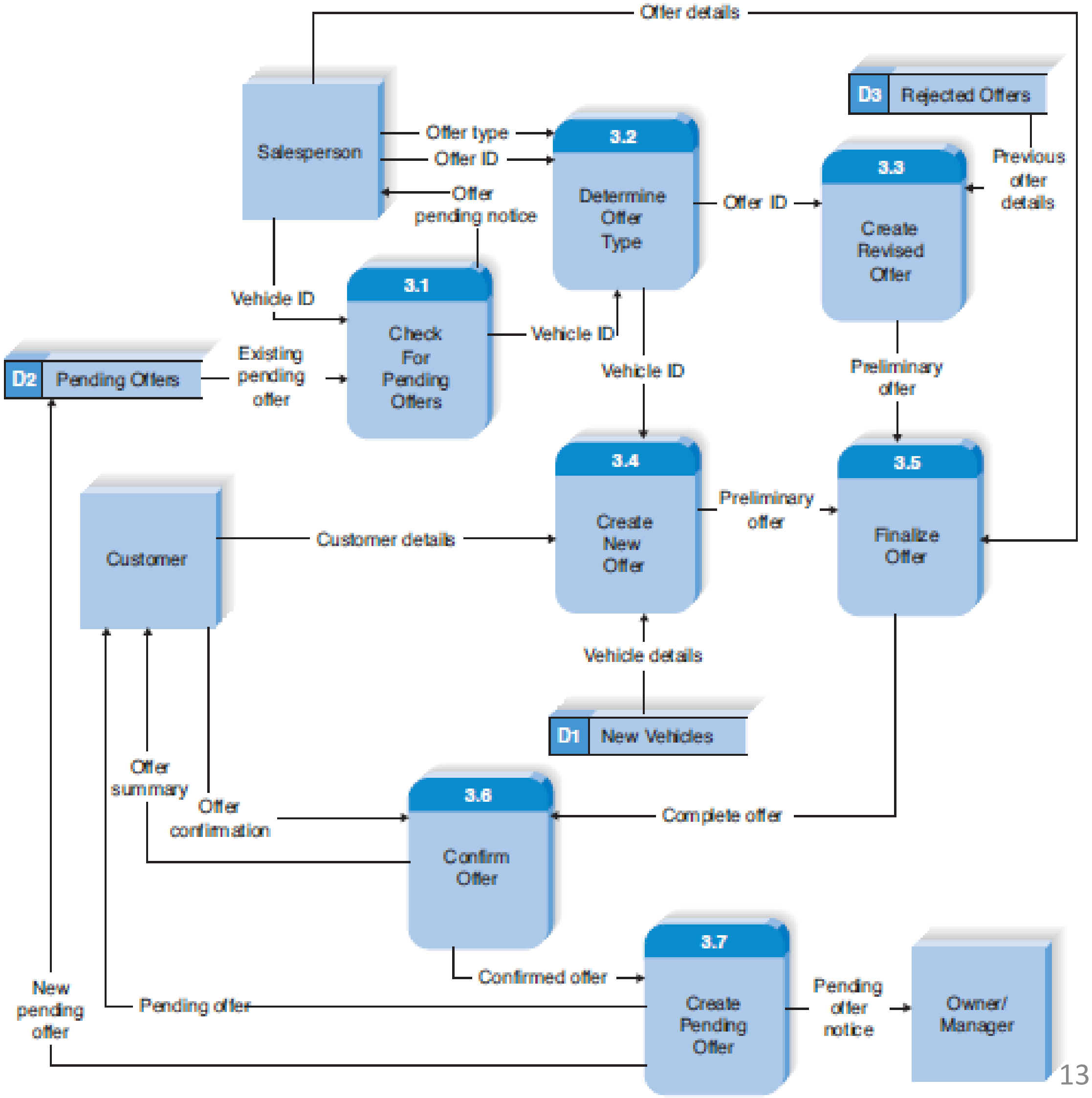
Creating DFD

- Level 0



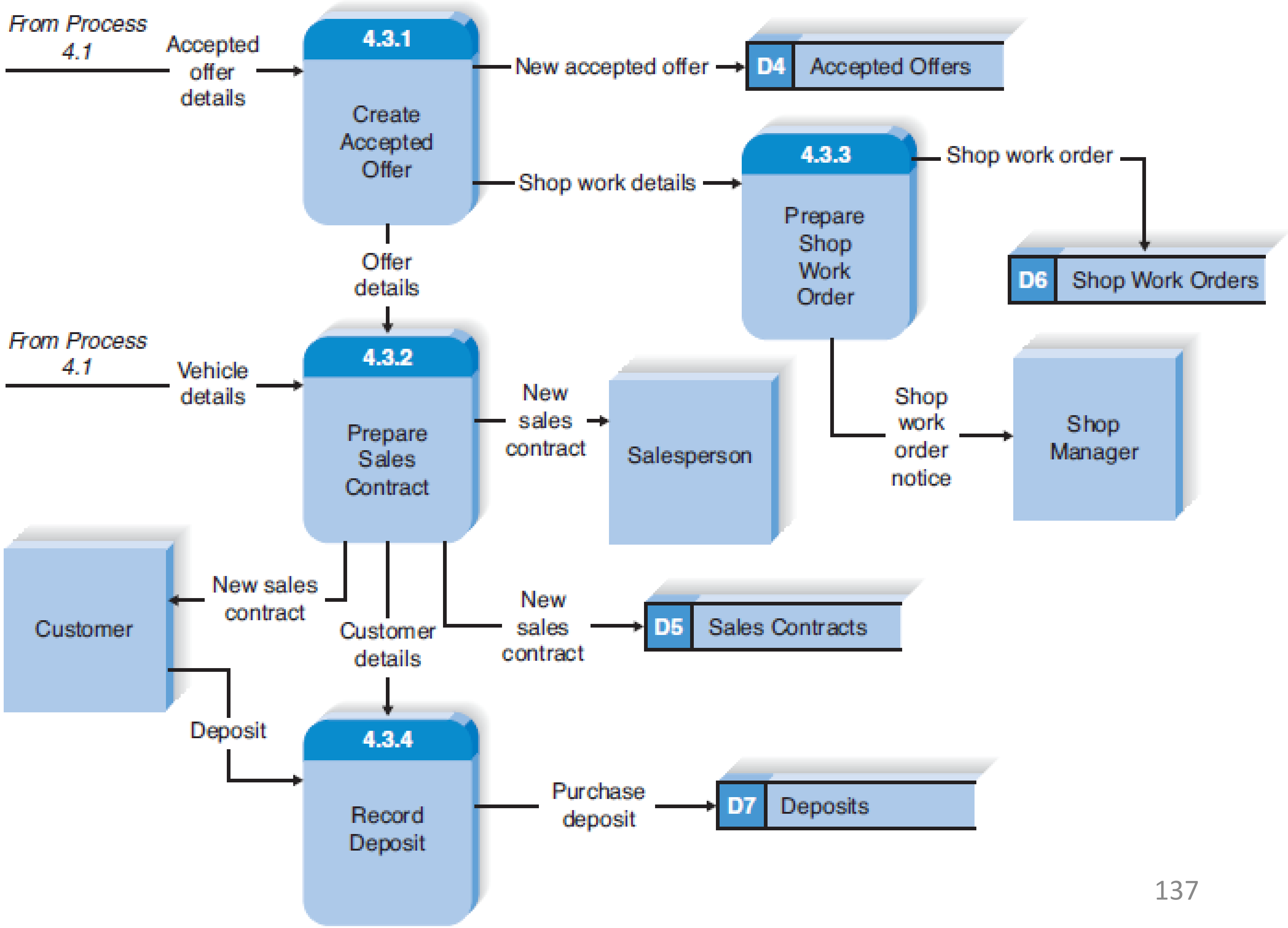
Creating DFD

- Level 1



Creating DFD

- Level 2

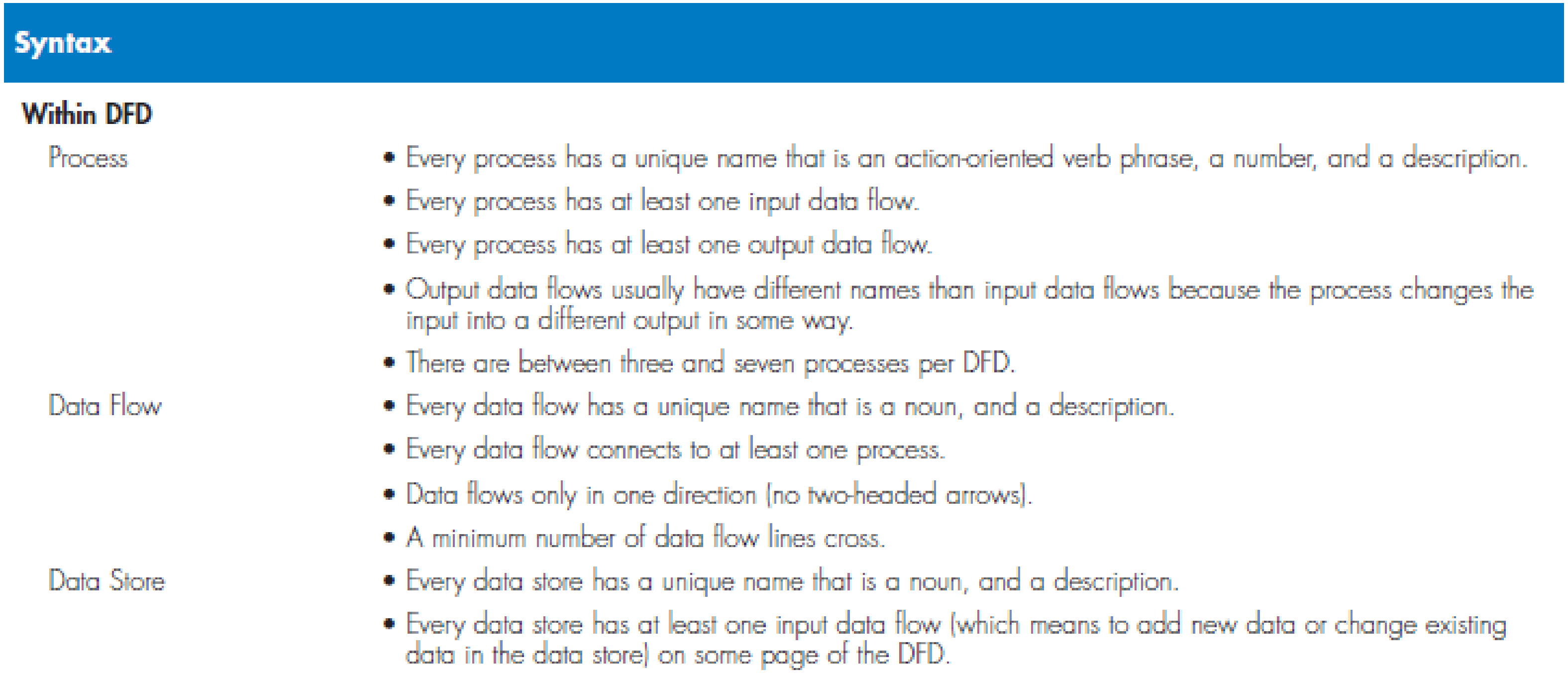


Validating DFD

- There are two fundamentally different types of problems that can occur in DFDs:
  - Syntax errors.
  - Semantics errors.

Validating DFD

- Examples for syntax errors:



Syntax	
Within DFD	
Process	• Every process has a unique name that is an action-oriented verb phrase, a number, and a description. • Every process has at least one input data flow. • Every process has at least one output data flow. • Output data flows usually have different names than input data flows because the process changes the input into a different output in some way. • There are between three and seven processes per DFD.
Data Flow	• Every data flow has a unique name that is a noun, and a description. • Every data flow connects to at least one process. • Data flows only in one direction (no two-headed arrows). • A minimum number of data flow lines cross.
Data Store	• Every data store has a unique name that is a noun, and a description. • Every data store has at least one input data flow (which means to add new data or change existing data in the data store) on some page of the DFD.

Validating DFD

- Semantic errors:
  - Semantics errors cause the most problems in system development.
  - They are much harder to find and fix because doing so requires a good understanding of the business process
- Checking:
  - Walk-through with users.
  - Role-play the process with users.

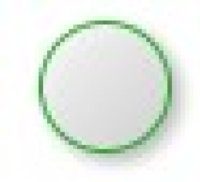
- One of the most subtle forms of semantics error occurs when a process creates an output, but has insufficient inputs to create it.

BPMN

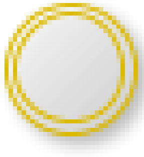
- Business Process Model and Notation
- http://www.bpmn.org
- defines a Business Process Diagram (BPD), which is based on a flowcharting technique tailored for creating graphical models of business process operations.
- It is a network of graphical objects, which are activities (i.e., work) and the flow controls that define their order of performance.

Basic elements

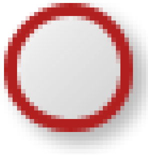
Main events



- ✓ Start



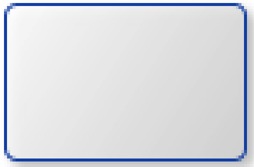
- ✓ Intermediate



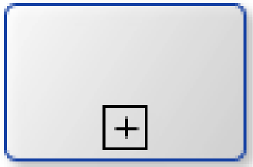
- ✓ End

Basic elements

Task



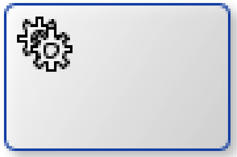
✓ Single task



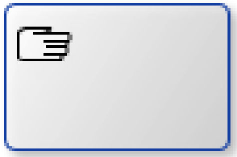
✓ Sub process

Basic elements

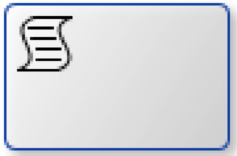
Detailed task



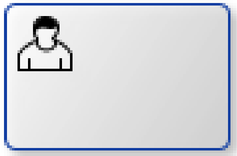
- ✓ Service



- ✓ Manual



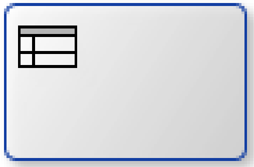
- ✓ Script



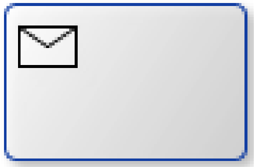
- ✓ User

Basic elements

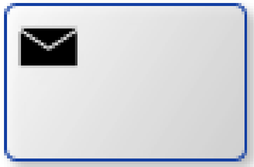
Detailed task



✓ Business rule



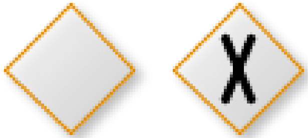
✓ Receive



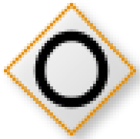
✓ Send

Basic elements

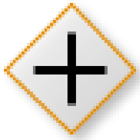
- Gateway



✓ Exclusive



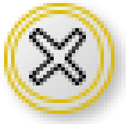
✓ Inclusive



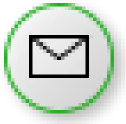
✓ Parallel

Basic elements

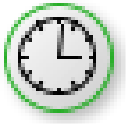
Event



- ✓ Cancel



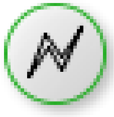
- ✓ Message



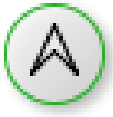
- ✓ Timer

Basic elements

Event



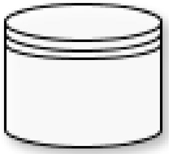
- ✓ Error



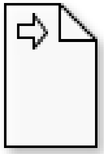
- ✓ Escalation

Basic elements

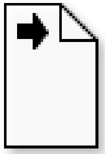
Data



✓ Store



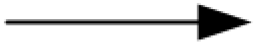
✓ Input



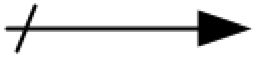
✓ Output

Basic elements

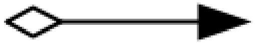
Flow



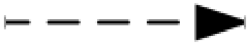
✓ Sequence



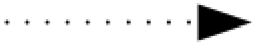
✓ Default sequence



✓Conditional sequence



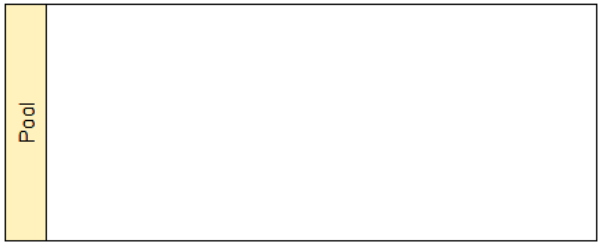
✓ Message



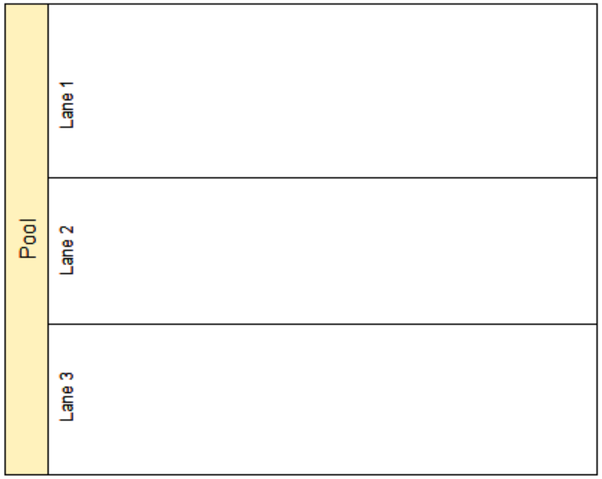
✓ Association

Pools and lanes

Pool



- ✓ Empty



- ✓ With lanes

Rules

Sequence flow

- ✓ Are used to show the order that Activities will be performed in a Process
- ✓ Cannot cross sub process boundaries
- ✓ Cannot cross pool boundaries

Rules

Message flow

- ✓ Are used to show communication between Participants
- ✓ They cannot connect objects that are within the same Pool

Rules

Gateways

- ✓ Gateway are not decisions
  - ✓ Gateways do not make decisions they only direct the flow
  - ✓ A decision outcome should be determined in an Activity prior to the Gateway
- ✓ Number of flows should be equal in pairs

Rules

Tasks

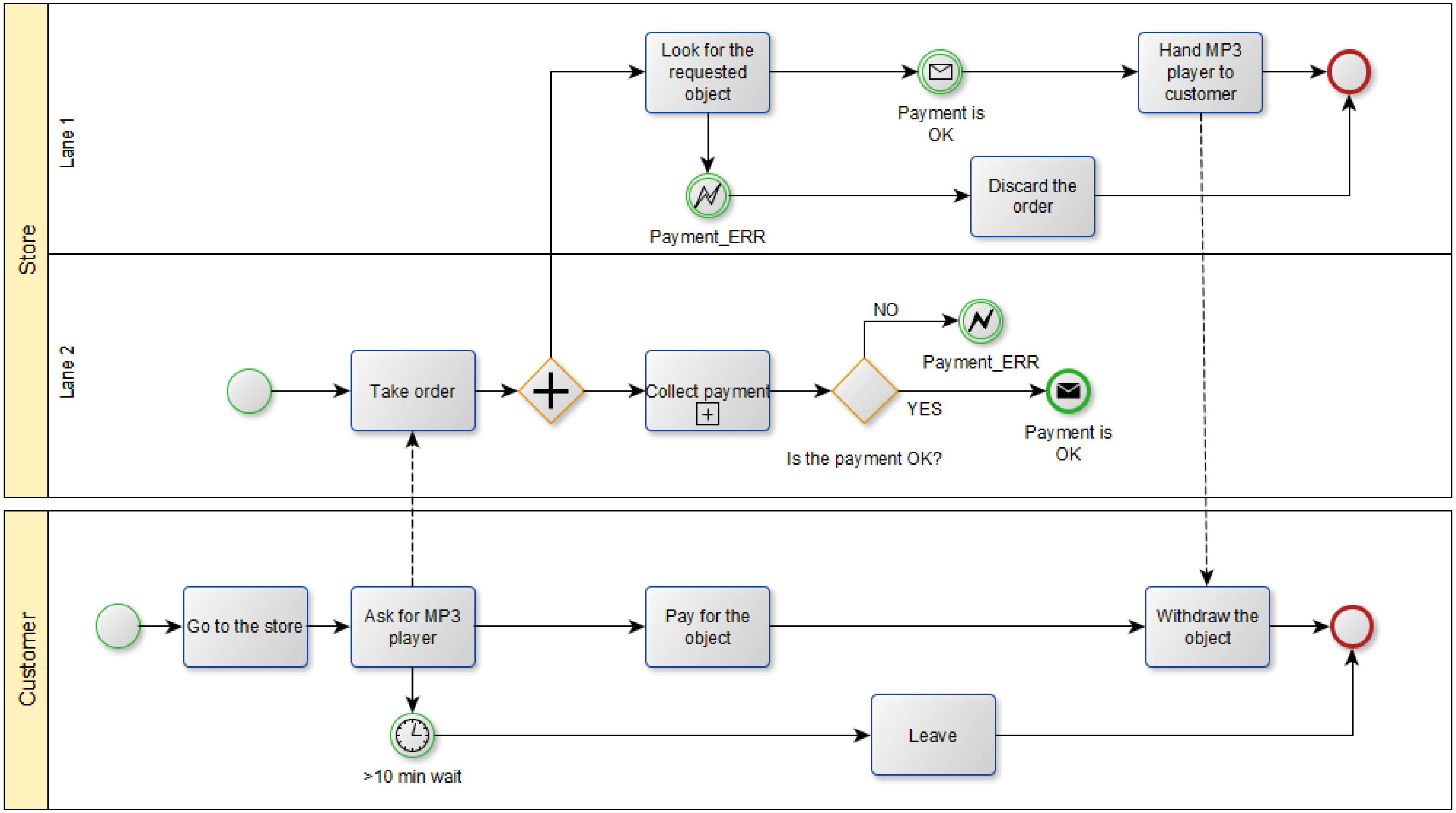
- ✓ Use a Manual Task to depict work effort expected to be performed without the aid of any software application
- ✓ Use a User Task to depict semi-automated work effort where a human performer uses a software application to complete the Task
- ✓ Use a Service Task to depict an automated work effort

Rules

Messages

- ✓ Use a Message Event if the sending or receiving of the Message is considered instantaneous
- ✓ Message Task if the sending or receiving of the Message can be interrupted
- ✓ From a temporal perspective; an Event maps to a time point on a time line and a Task maps to a time interval

Example


Store
Lane 1
Look for the requested object
Payment is OK
Hand MP3 player to customer
Payment_ERR
Discard the order
Lane 2
Take order
Collect payment
NO
Payment_ERR
YES
Payment is OK
Is the payment OK?
Customer
Go to the store
Ask for MP3 player
Pay for the object
Withdraw the object
Leave
>10 min wait

THANKS FOR ATTENTION!

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

EFOP-3.4.3-16-2016-00009

A felsőfokú oktatás minőségének és hozzáférhetőségének együttes javítása a Pannon Egyetemen

SYSTEM ANALYSIS AND SYSTEM DESIGN

6. DATA MODELLING

Imre Budai

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

Objectives

- Explain the rules and style guidelines for creating entity relationship diagrams.
- Create an entity relationship diagram.
- Describe the use of a data dictionary and metadata.
- Explain how to balance entity relationship diagrams and data flow diagrams.
- Describe the process of normalization.

Introduction

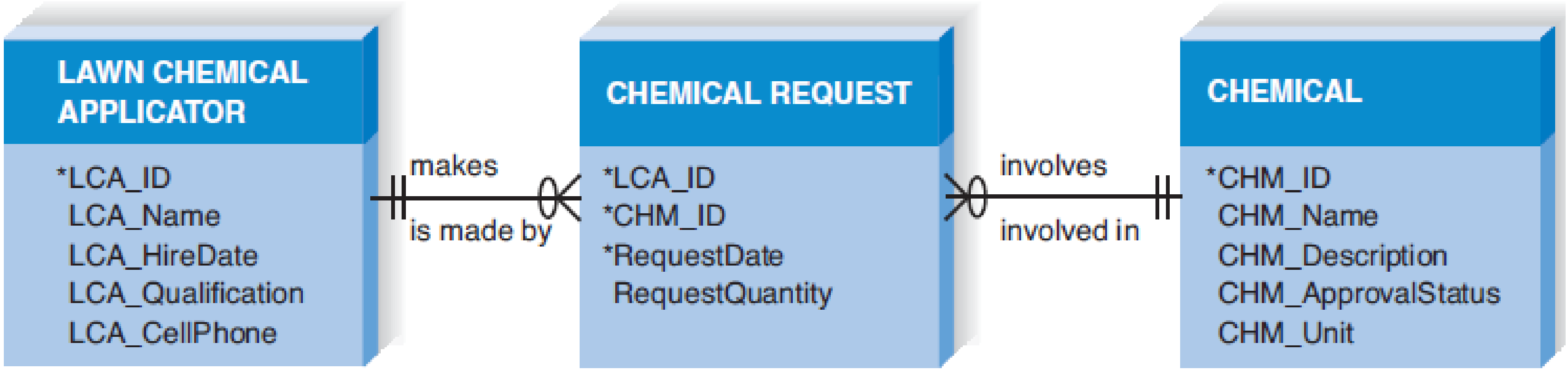
- A data model is a formal way of representing the data that are used and created by a business system.
  - it illustrates people, places, or things about which information is captured
  - And how they are related to each other.
- The data model is drawn by an iterative process in which the model becomes more detailed and less conceptual over time.

Entity Relationship Diagram

- An entity relationship diagram (ERD) is a picture which shows the information that is created, stored, and used by a business system.
- On an ERD:
  - Similar kinds of information are listed together and placed inside boxes called entities.
  - Lines are drawn between entities to represent relationships among the data.
  - Special symbols are added to the diagram to communicate high-level business rules that need to be supported by the system.

Example

- Lawn Chemical Applicator



Entity Relationship Diagram

- The ERD also communicates high-level business rules.
  - “A customer may place many orders.”
    - For e.g.: We can see by the two bars placed on the line closest to the LCA that a Chemical Request is made by exactly one LCA.
    - The system should not permit a chemical request to be made that does not involve an LCA.

Elements

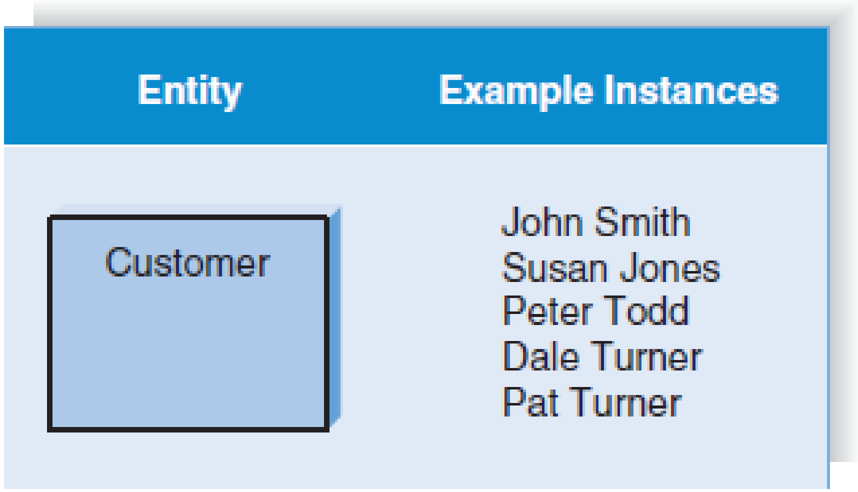
- Following elements can be used>
  - Entity
  - Attribute
  - Relationship
  - Cardinality
  - Modality

Entity

- The entity is the basic building block for a data model.
- It is a person, place, event, or thing about which data is collected.
  - For e.g.: an employee, an order, or a product.
- All entities have
  - Name
  - Short description
  - Identifier
- Entities represent something for which there exist multiple instances, or occurrences.

Entity

- For e.g.: For example, John Smith and Susan Jones could be instances of the customer entity.
- We would expect the customer entity to stand for all of the people with whom we have done business.
- Each of them would be an instance in the customer entity.



Entity	Example Instances
Customer	John Smith Susan Jones Peter Todd Dale Turner Pat Turner

Attribute

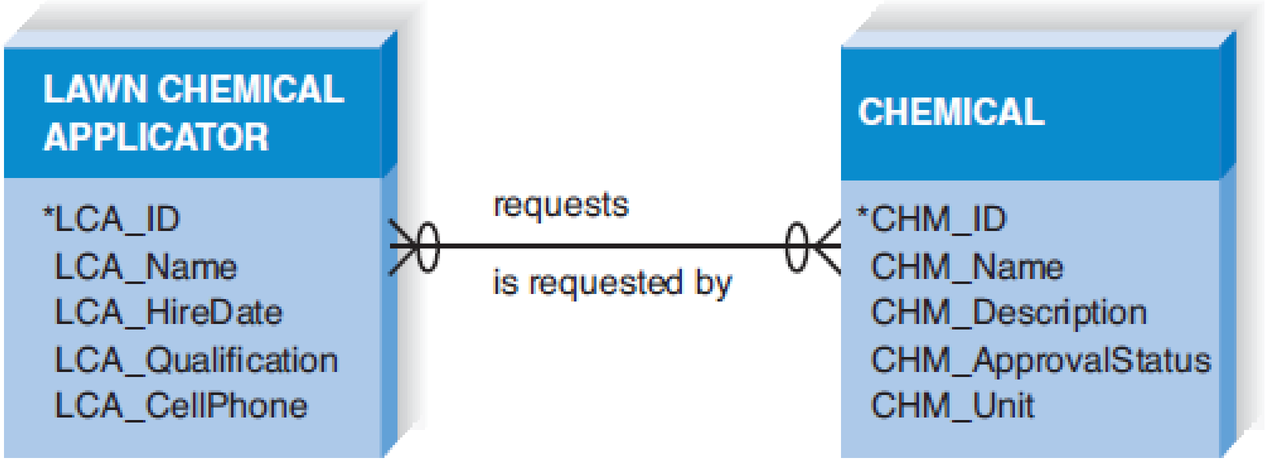
- An attribute is some type of information that is captured about an entity.
  - For e.g.: last name, home address, and e-mail address are all attributes of a customer.
  - *Only those that actually will be used by a business process should be included in the model.*
- Attributes are nouns that are listed within an entity.
- Usually, some form of the entity name is appended to the beginning of each attribute to make it clear as to what entity it belongs.
- One or more attributes can serve as the identifier

Relationship

- Relationships are associations between entities.
- They are shown by lines that connect the entities together.
- Every relationship has a parent entity and a child entity.
  - The parent being the first entity in the relationship, and the child being the second.
- Relationships should be clearly labelled with active verbs so that the connections between entities can be understood. If one verb is given to each relationship, it is read in two directions.

Relationship

- An LCA makes a chemical request.
- A chemical request is made by an LCA.
- The top words are read from parent to child, and the bottom words are read from child to parent..



Cardinality

- It is a the ratio of parent instances to child instances.
- To determine the cardinality for a relationship, we ask ourselves:
  - “How many instances of one entity are associated with an instance of the other?

    For example, an LCA makes how many chemical requests?
- It can be: 1:1, 1:N, or M:N
- 1:1 – “one to one”
  - One instance of the parent entity is associated with one instance of the child entity.

    One reserved parking place *is assigned* ***to each honoured employee****, and* ***each honoured employee*** *is assigned* ***one reserved parking place****.*

Cardinality

- 1:N – “one to many”
  - A single instance of a parent entity is associated with many instances of a child entity.
  - The child entity instance is related to only one instance of the parent.
  - For e.g.:
    - An LCA (parent entity) can make many Chemical Requests (child entity), but
    - A particular Chemical Request is made by only one LCA.
- M:N – “many to many”
  - Many instances of a parent entity can relate to many instances of a child entity.

Modality

- Relationships have a modality of null or not null.
- Which refers to whether or not an instance of a child entity can exist without a related instance in the parent entity.

  The modality of a relationship indicates whether the child-entity instance is required to participate in the relationship.

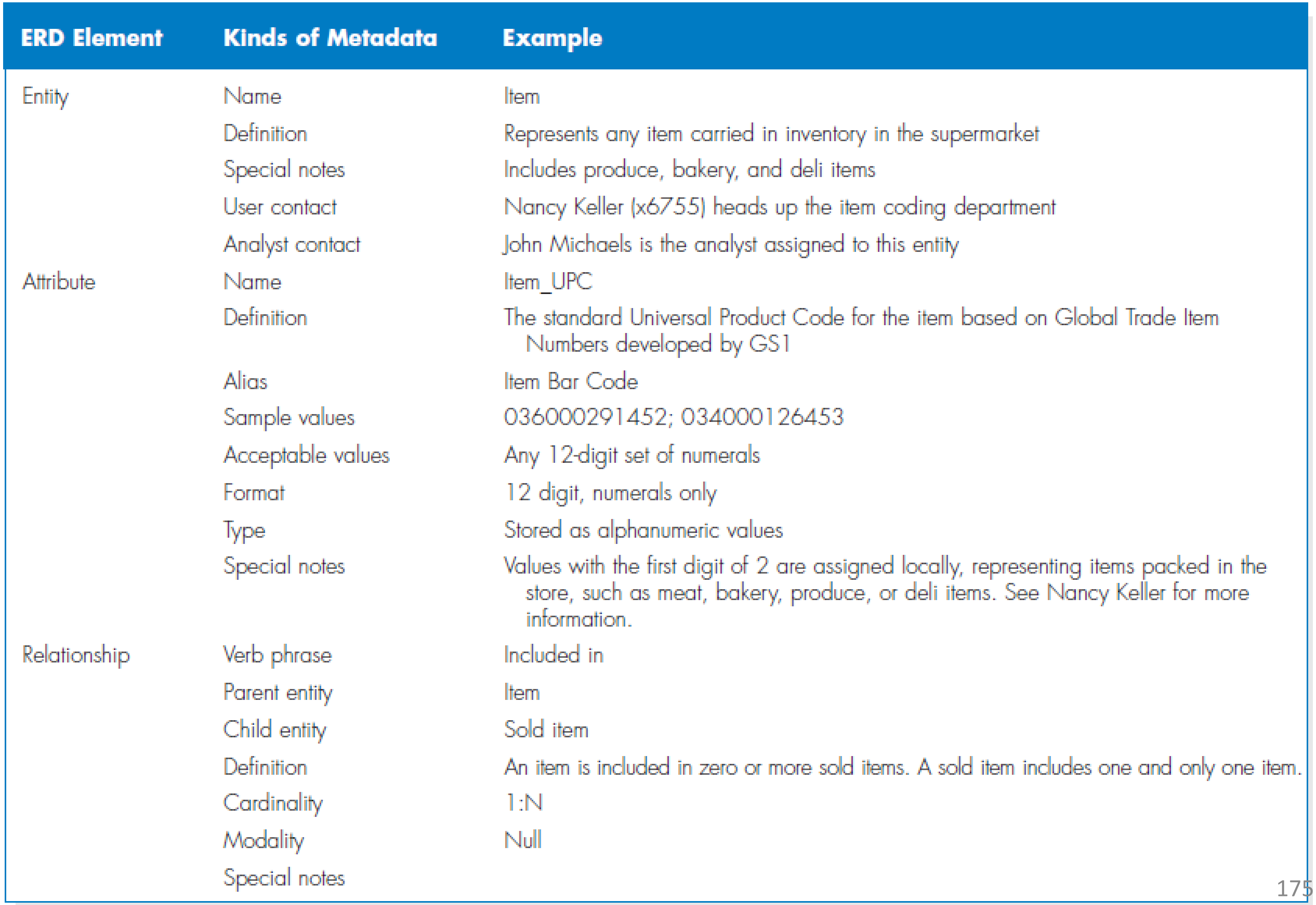
  - *Can you have a Chemical Request without a Chemical?*
  - *Can you have a Chemical without a Chemical Request?*

Data Dictionary Metadata

- CASE tool is used to help build ERDs.
- Every CASE tool has something called a data dictionary, which quite literally is where the analyst goes to define or look up information about the entities, attributes, and relationships. (Like MS Visio for e.g.)
- The information you see in the data dictionary is called metadata.
  - *It is anything that describes an entity, attribute, or relationship.*

    entity names, attribute descriptions, and relationship cardinality

Data Dictionary Metadata



ERD Element	Kinds of Metadata	Example
Entity	Name	Item
	Definition	Represents any item carried in inventory in the supermarket
	Special notes	Includes produce, bakery, and deli items
	User contact	Nancy Keller (x6755) heads up the item coding department
	Analyst contact	John Michaels is the analyst assigned to this entity
Attribute	Name	Item_UPC
	Definition	The standard Universal Product Code for the item based on Global Trade Item Numbers developed by GS1
	Alias	Item Bar Code
	Sample values	036000291452; 034000126453
	Acceptable values	Any 12-digit set of numerals
	Format	12 digit, numerals only
	Type	Stored as alphanumeric values
	Special notes	Values with the first digit of 2 are assigned locally, representing items packed in the store, such as meat, bakery, produce, or deli items. See Nancy Keller for more information.
Relationship	Verb phrase	Included in
	Parent entity	Item
	Child entity	Sold item
	Definition	An item is included in zero or more sold items. A sold item includes one and only one item.
	Cardinality	1:N
	Modality	Null
	Special notes	

Data Dictionary Metadata

- Metadata are stored in the data dictionary so that they can be shared and accessed by developers and users throughout the SDLC.
- It allows you to record the standard pieces of information about your elements in one place.
- It makes that information accessible to many parts of a project:
  - Process model
  - Data flows
  - User interface
- Every change in the data dictionary, the change ripples to the relevant parts of the project that are affected.

Creating ERD

1. Identify the Entities

   The entities should represent the major categories of information that you need to store in your system.

   In Use Case: the major inputs, major outputs, and the information used for the use case steps

2. Add Attributes and Assign Identifiers

   The information that describes each entity becomes its attributes.

   From: use case, functional/data requirements, interviews, document analysis

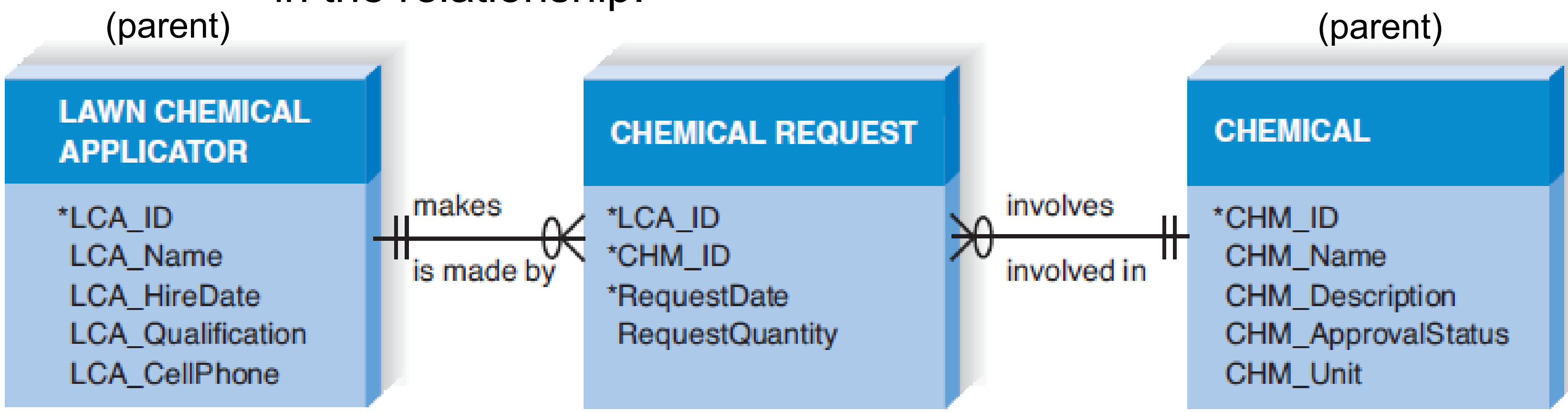
   - When finished one or more of them will become the entity’s identifier.
   - The identifier must be an attribute(s) that is able to uniquely identify a single instance of the entity.

Creating ERD

3. Identify Relationships

The last step is to determine how the entities are related to each other.

- The easiest approach: is to begin with one entity and determine all the entities with which it shares relationships.
- When you find a relationship to include on the model, you need to determine its cardinality and modality.
  - For cardinality, ask how many instances of each entity participate in the relationship.



(parent)

LAWN CHEMICAL APPLICATOR

*LCA_ID
LCA_Name
LCA_HireDate
LCA_Qualification
LCA_CellPhone

makes

is made by

CHEMICAL REQUEST

*LCA_ID
*CHM_ID
*RequestDate
RequestQuantity

involves

involved in

(parent)

CHEMICAL

*CHM_ID
CHM_Name
CHM_Description
CHM_ApprovalStatus
CHM_Unit

Advanced Syntax

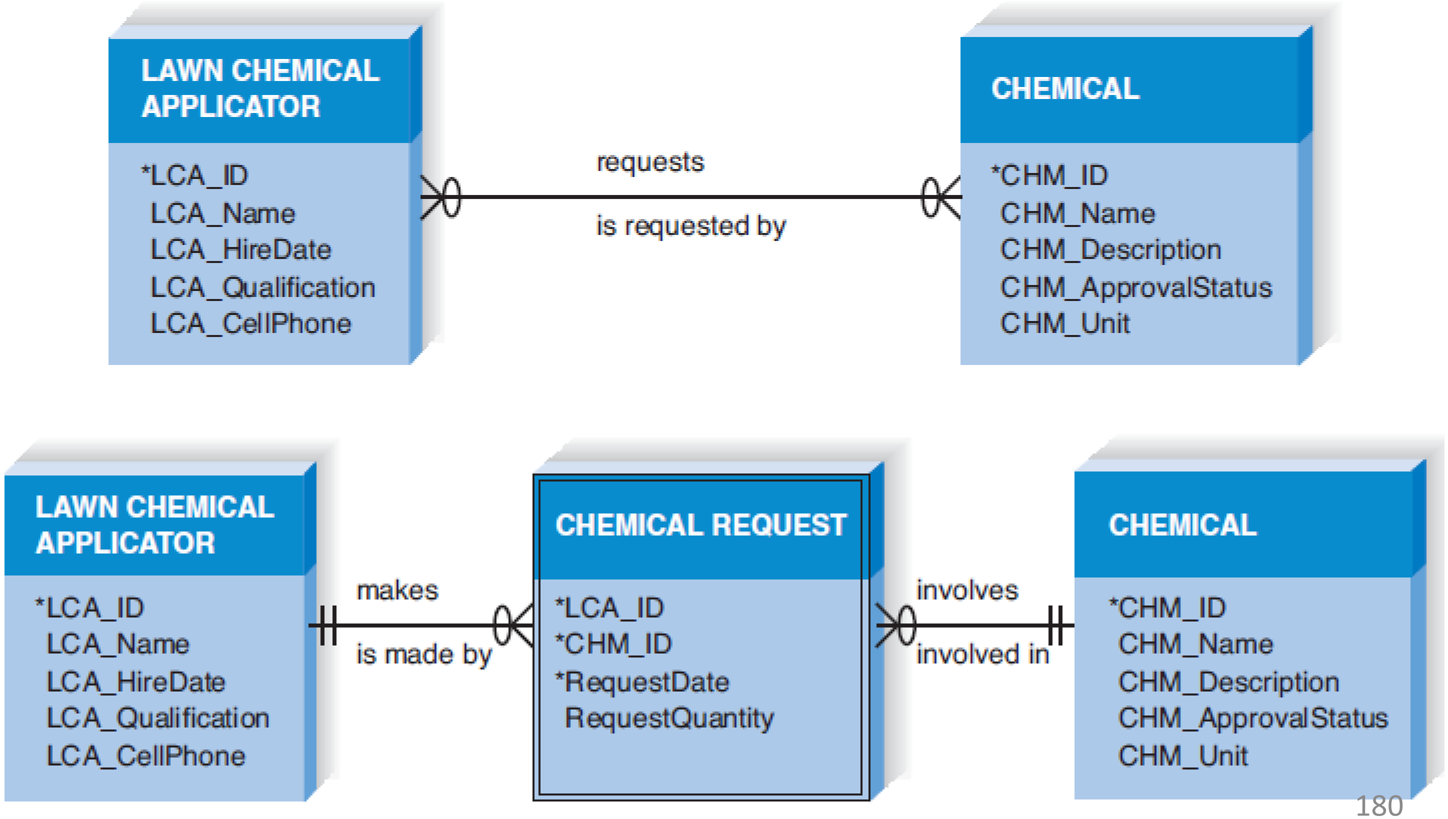
- Independent Entity
  - It can exist without the help of another entity.
  - Its identifiers were created from their own attributes.

    Attributes from other entities were not needed to uniquely identify instances of these entities.
- Dependent Entity
  - It is when a child entity does require attributes from the parent entity to uniquely identify an instance.

    Marked with double border line.

Advanced Syntax

- Intersection Entity
  - It exists in order to capture some information about the relationship between two other entities.



Validation

- Normalization
  - It helps analysts with validating the model that they have drawn.
  - It has 3 normal forms that ensures a clear and unambiguous data structure.
- Balancing ERD with DFD
  - Process model contains two data components
    - Data flow (DFD)
    - Data store

  These components of the DFD need to balance with the ERD.

Normalization

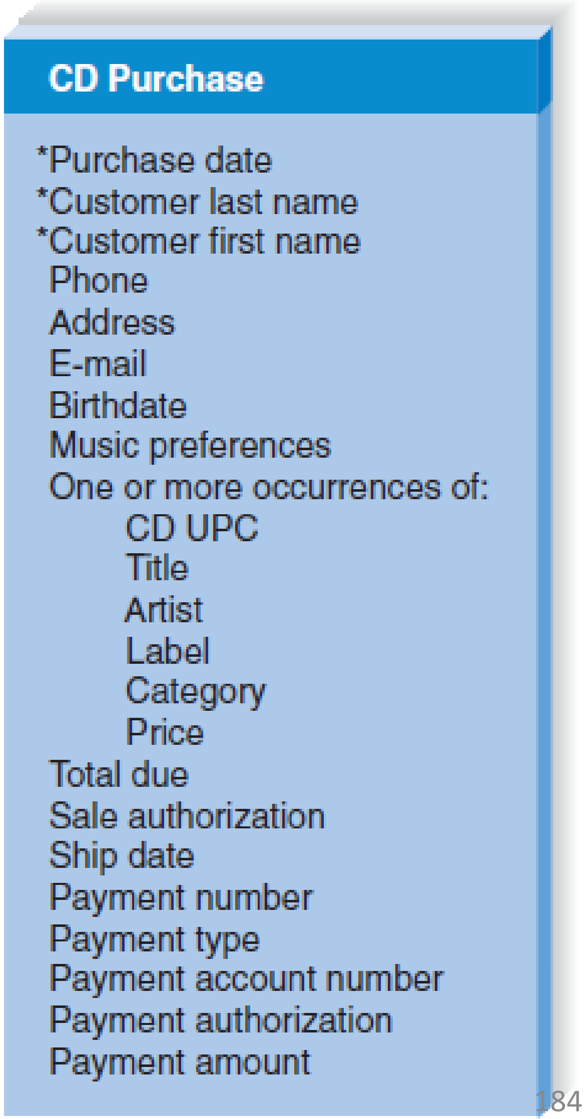
- First Normal Form (1NF)
  - It does not contain attributes that have repeating values for a single instance of an entity.
  - Every attribute in an entity should have only one value per instance for the model to “pass” 1NF.
- Second Normal Form (2NF)
  - The data model leads to entities containing attributes that are dependent on the whole identifier.
  - This means that the value of all attributes that serve as identifier can determine the value for all of the other attributes for an instance in an entity.

Normalization

- Third Normal Form (3NF)
  - In the resulting entities, none of the attributes is dependent on a non-identifier attribute (i.e., transitive dependency)

Example

- Digital Music Download system
  - Initial CD Sales System File
- Problem: There are two cases in which multiple values are captured for one or more attributes.
  - Multiple occurrences of CDs that are included in the purchase
  - The music preferences – many different preferences may be captured.



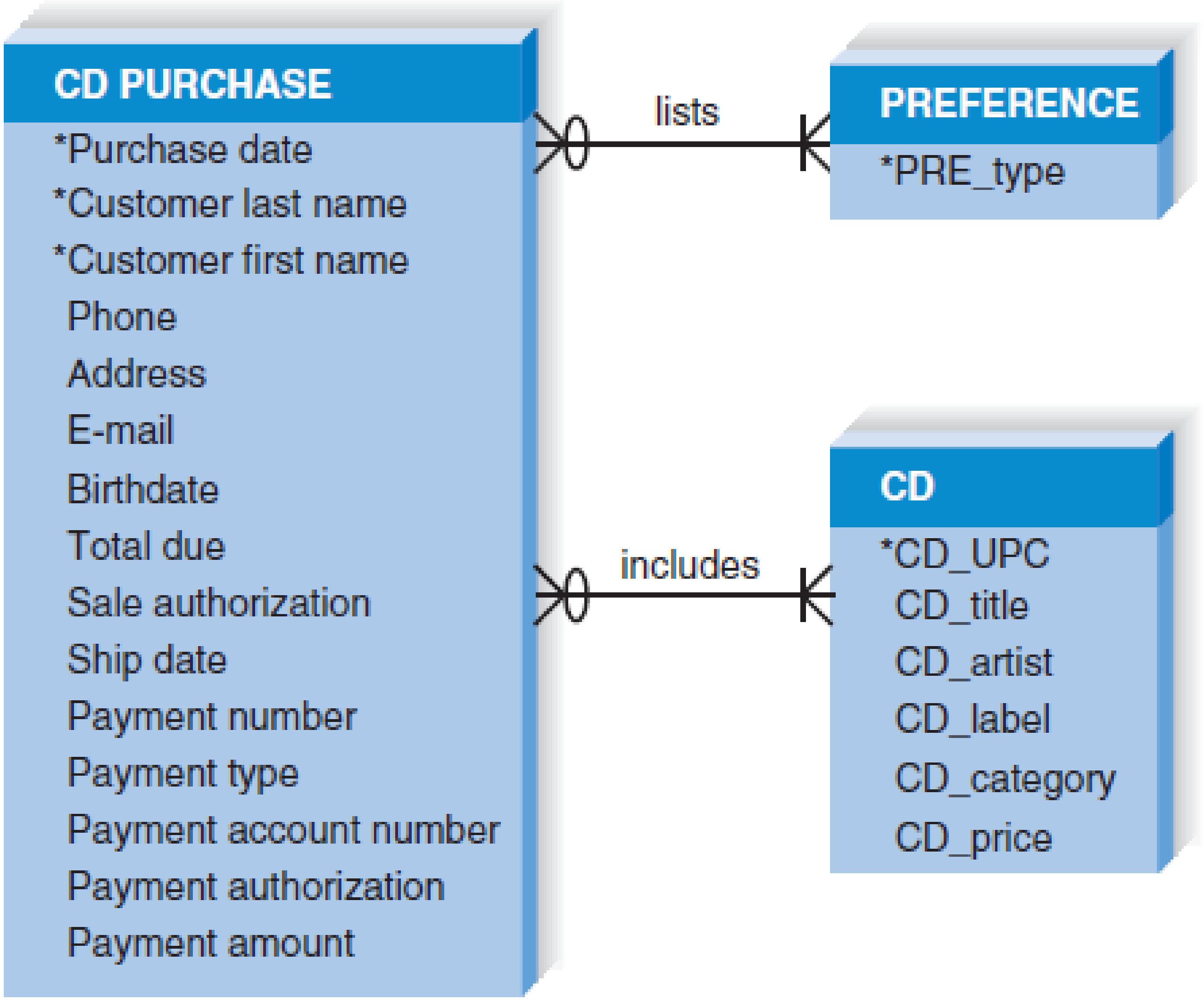
CD Purchase

*Purchase date
*Customer last name
*Customer first name
Phone
Address
E-mail
Birthdate
Music preferences
One or more occurrences of:
    CD UPC
    Title
    Artist
    Label
    Category
    Price
Total due
Sale authorization
Ship date
Payment number
Payment type
Payment account number
Payment authorization
Payment amount

Example

- The repeated group of attributes about each CD included in the purchase should be removed by creating a new entity called CD and placing all of the CD attributes into it.
- Create a new entity that contains preference information, and a relationship is added between CD purchase and preference.

Example



CD PURCHASE

*Purchase date
*Customer last name
*Customer first name
Phone
Address
E-mail
Birthdate
Total due
Sale authorization
Ship date
Payment number
Payment type
Payment account number
Payment authorization
Payment amount

PREFERENCE

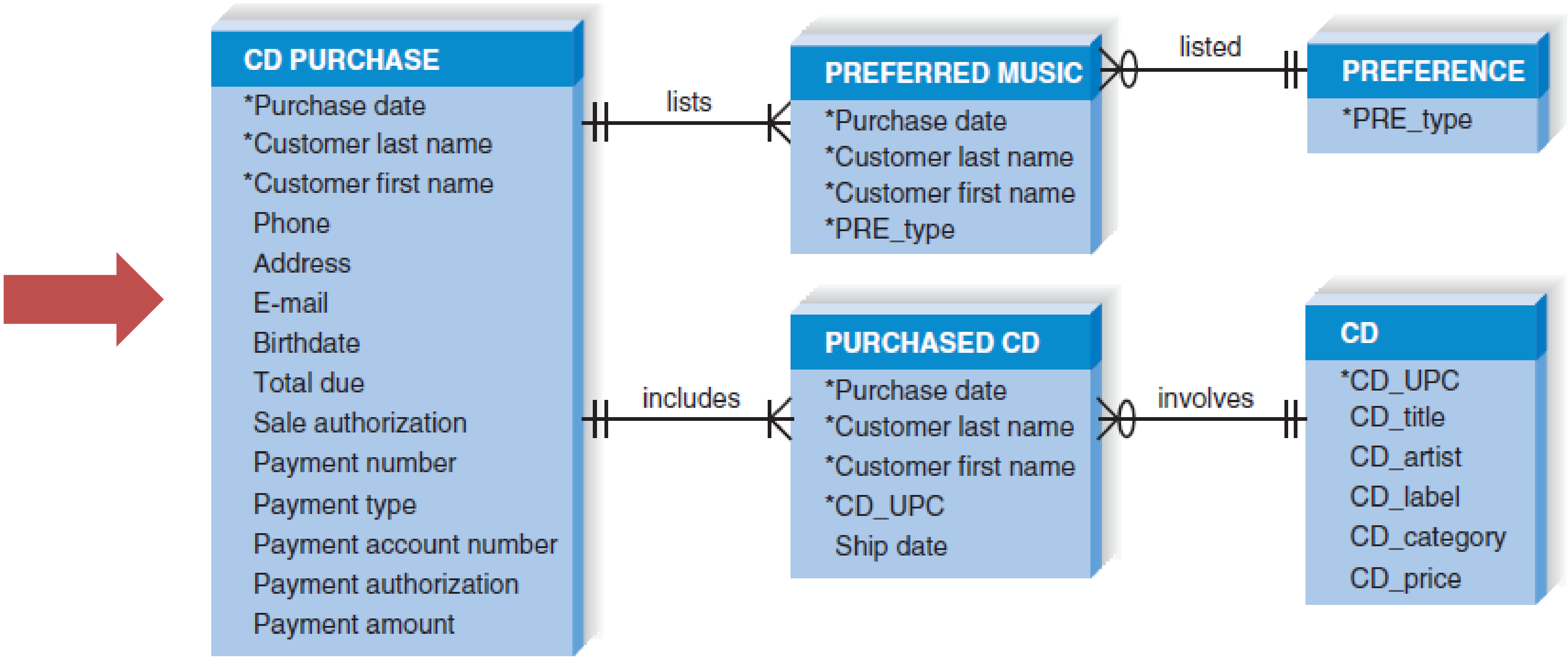
*PRE_type

CD

*CD_UPC
CD_title
CD_artist
CD_label
CD_category
CD_price

1NF

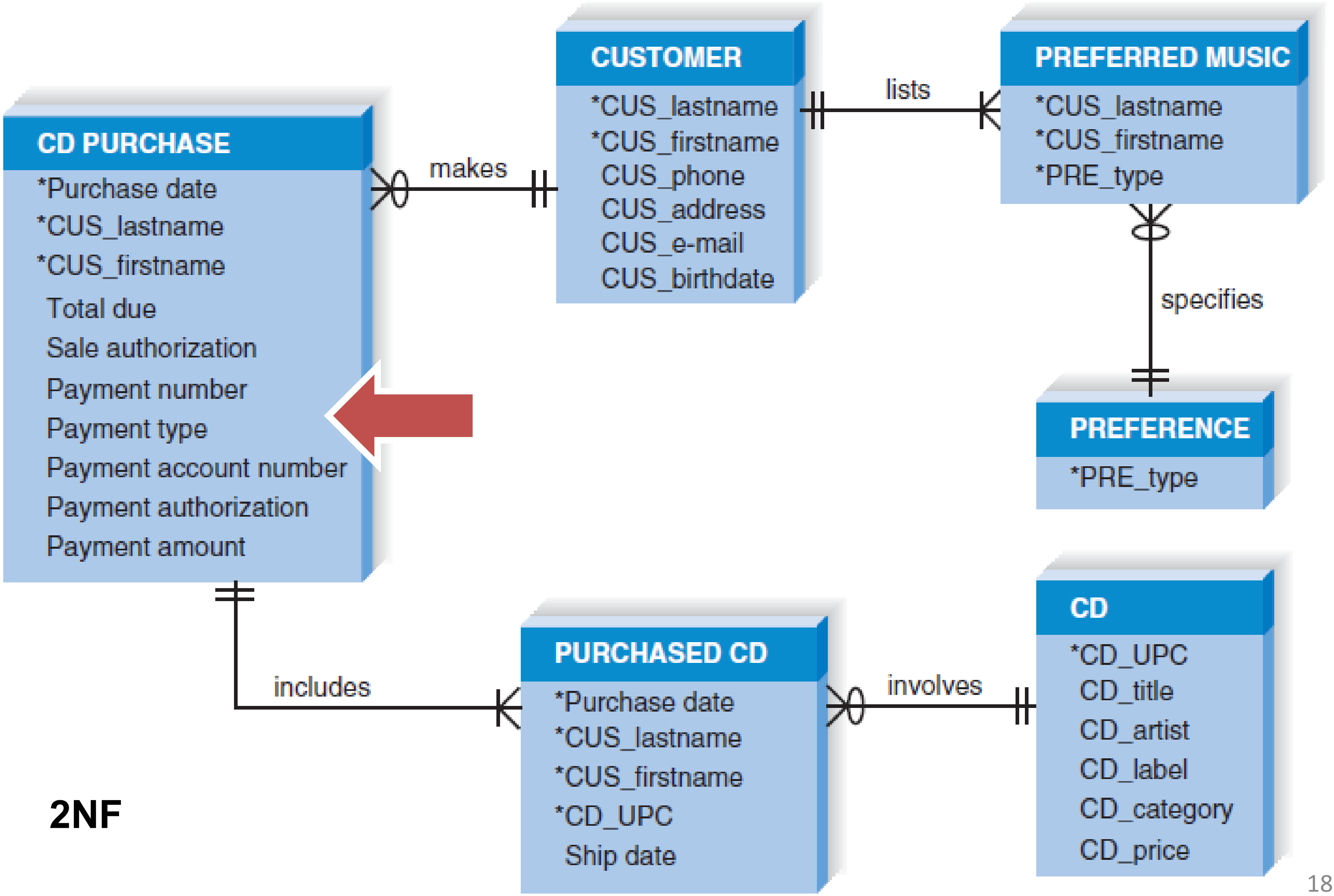
Example



- To resolve this problem, a new entity called customer was created, and the customer attributes were moved into the new entity.

A 1:N relationship exists between customer and CD purchase because a customer can purchase many CDs, but a CD purchase is associated with only one customer.

Example

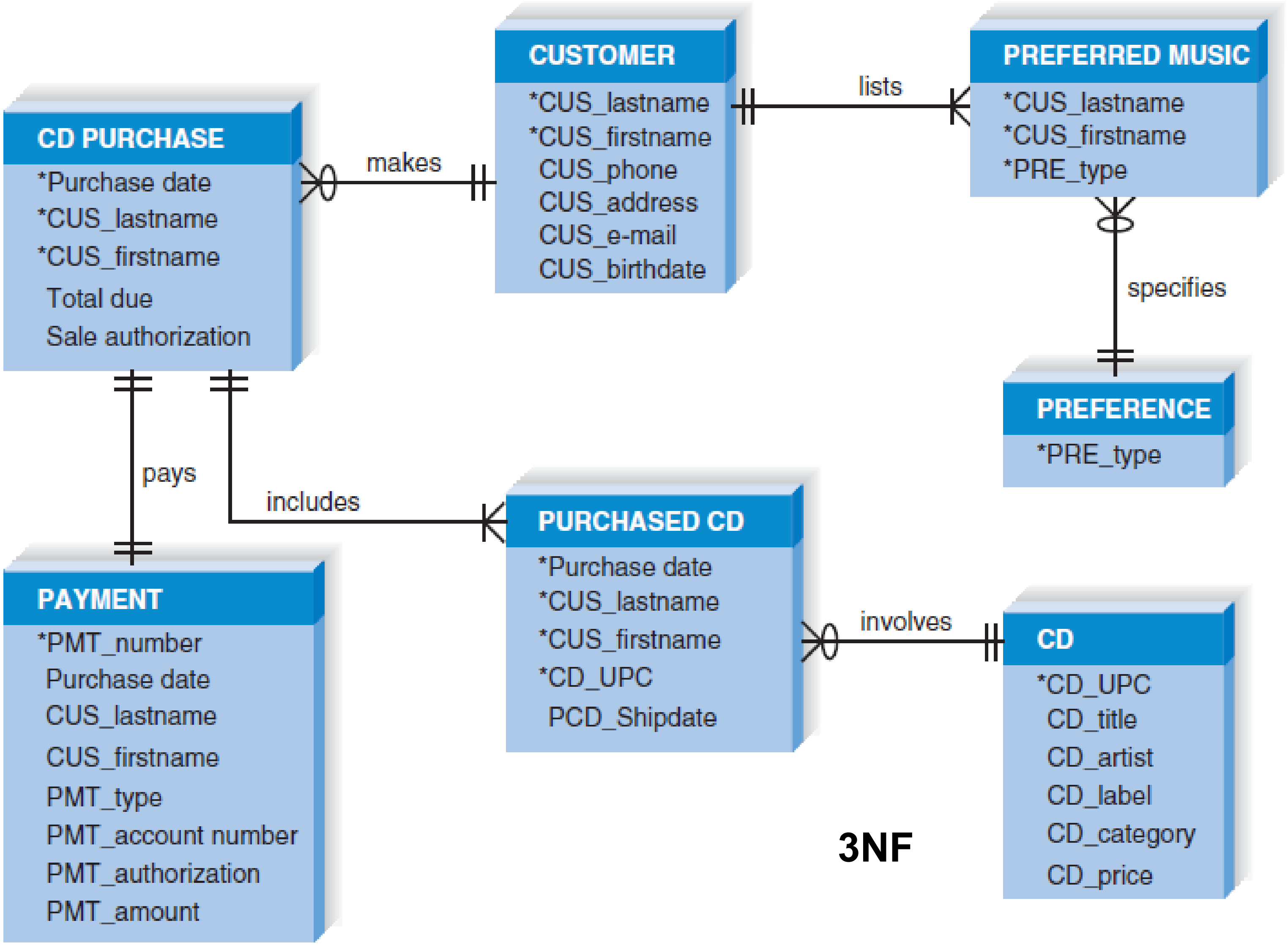

CD PURCHASE
*Purchase date
*CUS_lastname
*CUS_firstname
Total due
Sale authorization
Payment number
Payment type
Payment account number
Payment authorization
Payment amount
makes
CUSTOMER
*CUS_lastname
*CUS_firstname
CUS_phone
CUS_address
CUS_e-mail
CUS_birthdate
lists
PREFERRED MUSIC
*CUS_lastname
*CUS_firstname
*PRE_type
specifies
PREFERENCE
*PRE_type
includes
PURCHASED CD
*Purchase date
*CUS_lastname
*CUS_firstname
*CD_UPC
Ship date
involves
CD
*CD_UPC
CD_title
CD_artist
CD_label
CD_category
CD_price

2NF

Example

- The problem with the CD Purchase entity is that there are attributes in the entity that depend on the payment number, not the CD purchase date and customer first and last names.
- Create a separate payment entity and move the payment attributes to it.
    - The 1:1 relationship assumes that there is one payment for every CD purchase, and every CD purchase has one payment.
    - Also, a payment is required for every CD purchase, and every CD purchase requires a payment.

Example



CUSTOMER
*CUS_lastname
*CUS_firstname
CUS_phone
CUS_address
CUS_e-mail
CUS_birthdate

lists

PREFERRED MUSIC
*CUS_lastname
*CUS_firstname
*PRE_type

specifies

PREFERENCE
*PRE_type

CD PURCHASE
*Purchase date
*CUS_lastname
*CUS_firstname
Total due
Sale authorization

makes

pays

includes

PAYMENT
*PMT_number
Purchase date
CUS_lastname
CUS_firstname
PMT_type
PMT_account number
PMT_authorization
PMT_amount

PURCHASED CD
*Purchase date
*CUS_lastname
*CUS_firstname
*CD_UPC
PCD_Shipdate

involves

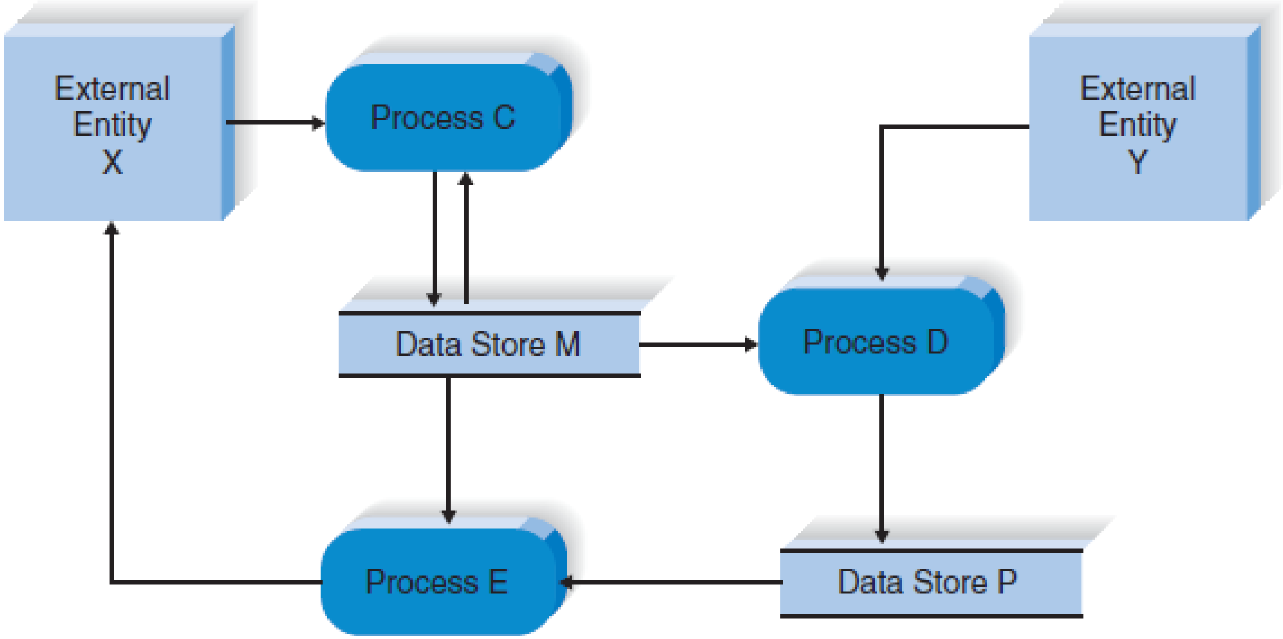
CD
*CD_UPC
CD_title
CD_artist
CD_label
CD_category
CD_price

3NF

Balancing with DFD

- Check your data model and see whether there are any entities you have created that do not appear as data stores on your process models.
- If there are, you should add them to your process models to reflect your decision to store information about that entity in your system.
- A useful tool to clearly depict the interrelationship between process and data models is the CRUD matrix.
    - It is helpful to develop the CRUD matrix on the basis of the logical process and data models and then revise it later in the design phase.

Balancing with DFD



	Process C	Process D	Process E
Data Entity M			
Attribute M-1	CRUD	R	R
Attribute M-2	CRUD		R
Attribute M-3	CRUD	R	
Attribute M-4	CRUD		R
Data Entity P			
Attribute P-1		C	R
Attribute P-2		C	
Attribute P-3		C	R

Balancing with DFD

- Check your data model and see whether there are any entities you have created that do not appear as data stores on your process models.
- If there are, you should add them to your process models to reflect your decision to store information about that entity in your system.
- A useful tool to clearly depict the interrelationship between process and data models is the CRUD matrix.
  - It is helpful to develop the CRUD matrix on the basis of the logical process and data models and then revise it later in the design phase.

Balancing with DFD

- We can identify places where attributes may have been omitted from the data stores/entities.
- In addition, we can verify that every attribute is created, read, updated, and deleted somewhere in the process model.
  - If it is not read by some process, then the attribute is probably not needed.
  - If it is not created or updated, the attribute probably needs to be added to a data flow(s) in the process model.

THANKS FOR ATTENTION!

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális
Alap

BEFEKTETÉS A JÖVŐBE

EFOP-3.4.3-16-2016-00009

A felsőfokú oktatás minőségének és hozzáférhetőségének együttes javítása a Pannon Egyetemen

SYSTEM ANALYSIS AND SYSTEM DESIGN

7. ARCHITECTURE DESIGN

Imre Budai

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

Objectives

- Describe the fundamental components of an information system.
- Describe server-based, client-based, and client–server architectures.
- Describe newer architectural options, such as cloud computing.
- Explain how operational, performance, security, cultural, and political requirements affect the architecture design.
- Create an architectural design.
- Create a hardware and software specification.

Introduction

- Designing the system architecture can be quite difficult.
- The key factors in architecture design
  - How the designers think about application architectures.
  - Describe the most common architecture used.
  - New developments in system architecture.
  - How the very general non-functional requirements from the analysis phase are refined into more specific requirements
  - How the requirements and architecture design can be used to develop the hardware and software specifications

Elements

- How the software components of the information system will be assigned to the hardware devices of the system.
- Major functions of the software.
- Major types of hardware onto which the software can be placed.

- There are numerous ways in which the software components can be placed on the hardware components, the most common architecture is the client–server architecture.

Elements

- Major architectural components of any system are the software and the hardware.
- The major software components of the system have to
  - Be identified and then
  - Allocated to the various hardware components on which the system will operate.

Elements

- All software systems can be divided into four basic functions.
  - Data storage
  - Data access logic
  - Application logic
  - Presentation logic
- There are three primary hardware components of a system
  - Client computers
  - Servers
  - Network

Client-Server Architectures

- There are numerous ways in which the software components can be placed on the hardware components, the most common architecture is the client–server architecture.
- attempt to balance the processing between client devices and one or more server devices.
- In these architectures
  - The client is responsible for the presentation logic
  - The server is responsible for the data access logic and data storage.

Client-Server Architectures

- The application logic may
  - Reside on the client
  - Reside on the server
  - Be split between both.
- Differentiation based on the application logic
  - Fat client: contains all or most of the application logic
  - Thin client: containing just a small portion of the application logic

Benefits

- Client–server architectures have four important benefits
  - Scalable.
  - Can support many different types of clients and servers.
  - (For thin client-server architectures) use Internet standards, it is simple to clearly separate the presentation logic, the application logic, and the data access logic and design each to be somewhat independent.
  - In case of server failure only the applications requiring that server will fail.

Middleware

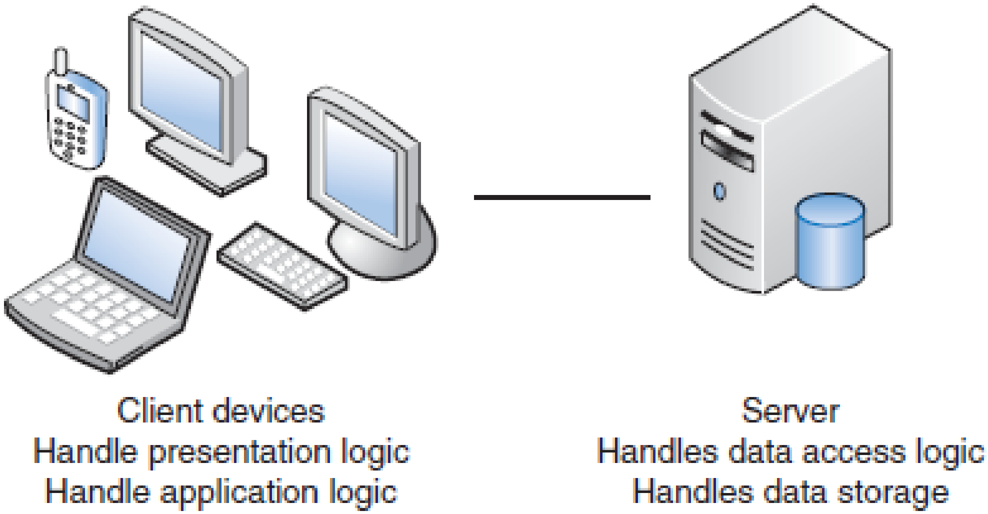
- Middleware is a type of system software designed to translate between different vendors' software.
- Middleware is installed on both the client computer and the server computer.
- The client software communicates with the middleware, which can reformat the message into a standard language that can be understood by the middleware, which assists the server software.

Limitations

- The most important is their complexity.
  - All applications in client–server computing have two parts, the software on the client side and the software on the server side.
  - Writing this software is more complicated than writing the traditional all-in-one software.
  - Updating the overall system with a new version of the software is more complicated.
    - All clients and servers must be updated.
    - It must be ensured that the updates are applied on all devices.

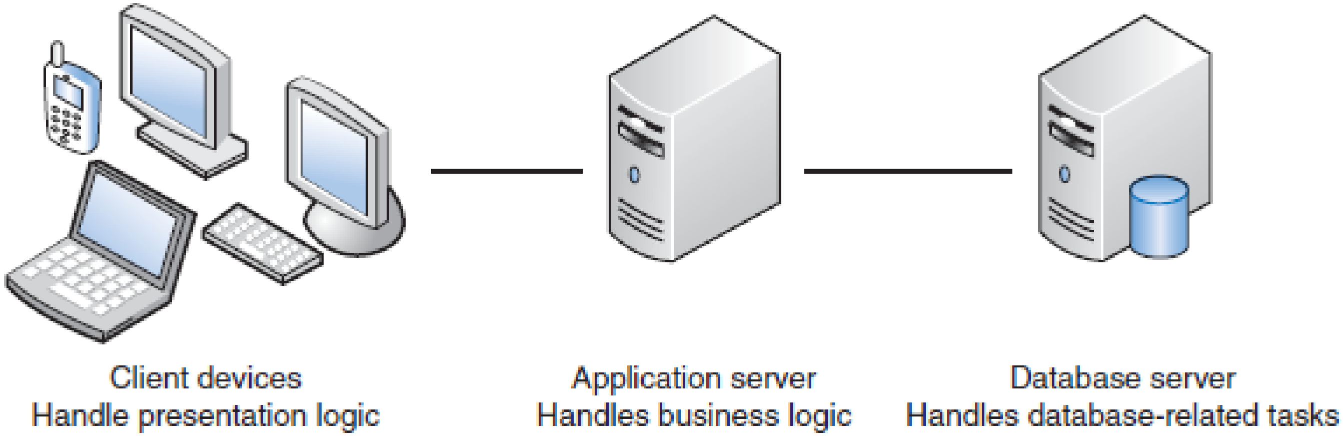
Tiers

- There are many ways in which the application logic can be partitioned between the client and the server
- Two-tiered architecture
  - The server is responsible for the data
  - The client is responsible for the application and presentation



Tiers

- Three-tiered architecture
  - The client computer is responsible for presentation logic
  - An application server(s) is responsible for the application logic
  - A separate database server(s) is responsible for the data access logic and data storage



Tiers

- The user interface runs on a desktop PC or workstation and uses a standard graphical user interface.
- The application logic may consist of one or more separate modules running on a workstation or application server.
- A relational DBMS running on a database server contains the data access logic and data storage.

Tiers

- N-tiered architecture
  - It distributes the work of the application (the middle tier) among multiple layers of more specialized server computers.
- It is common in today’s Web-based e-commerce systems.
  - The browser software on client computers makes HTTP requests to view pages from the Web server(s). The Web server(s) enable the user to view merchandise for sale by responding with HTML documents.
  - The shops components on the application server(s) are called as needed.

Tiers

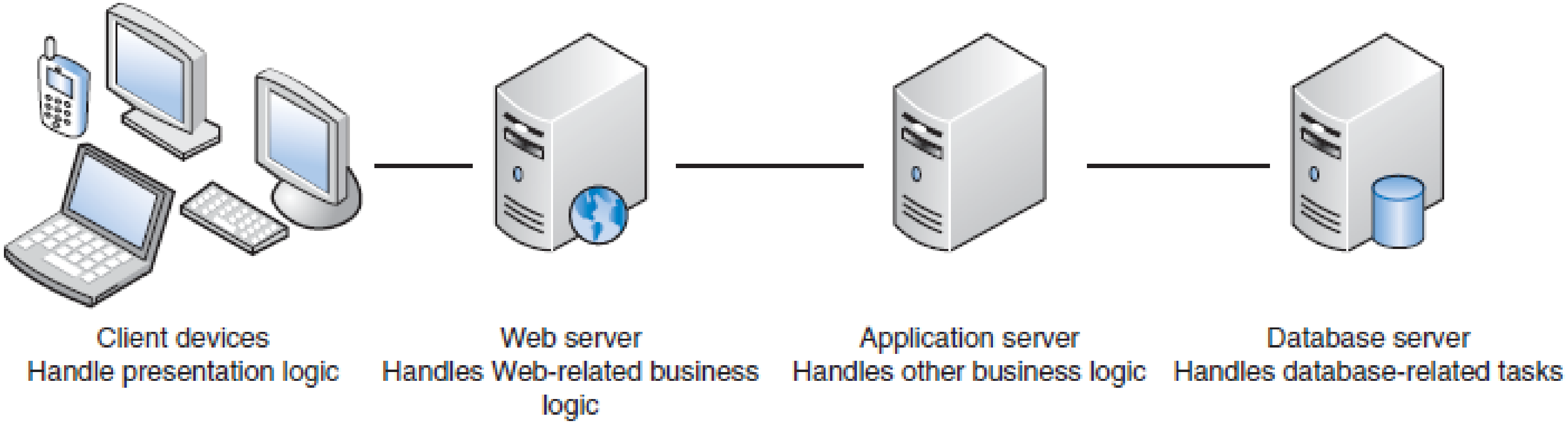
- It allows the user to
  - Put items in a shopping cart
  - Determine item pricing and availability
  - Compute purchase costs, sales tax, and shipping costs
  - Authorize payments
  - etc.
- These elements of business logic, or detailed processing, are stored on the application server(s).
- They are accessible to any application

Tiers

- For example, the cash register application that needs item price look-ups could use the same price determination business logic that is used by the e-commerce Web site.
- The modular business logic can be used by multiple, independent applications that need that particular business logic.
- The database server(s) manage the data components of the system.
- It makes easy to spread the different components on different servers and to partition the application logic on a Web-oriented server and a business-oriented server.

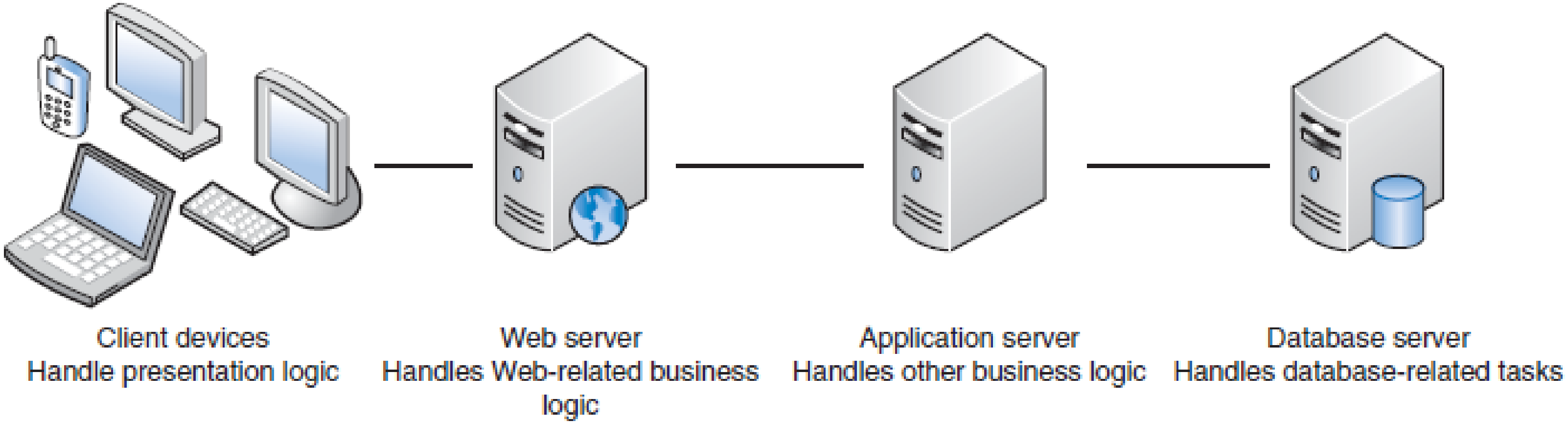
Tiers

- The primary advantage of an n-tiered client–server architecture compared with a two-tiered architecture is that it separates out the processing that occurs to better balance the load on the different servers.
- It is more scalable.


Client devices
Handle presentation logic
Web server
Handles Web-related business logic
Application server
Handles other business logic
Database server
Handles database-related tasks

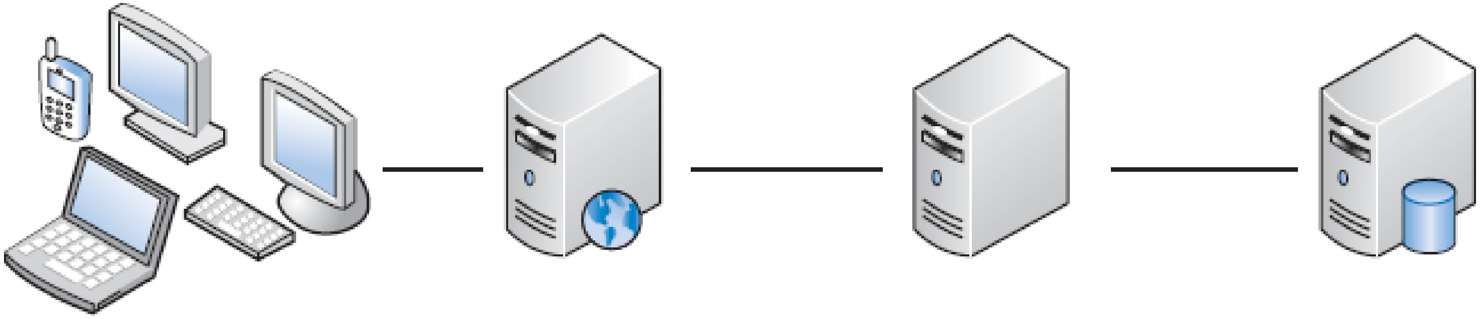
Tiers

- Performance issues (for e.g.)
  - Application server is too heavily loaded.
  - Database server is underused.


Client devices
Handle presentation logic
Web server
Handles Web-related business logic
Application server
Handles other business logic
Database server
Handles database-related tasks

Tiers

- Two primary disadvantages
  - First, the configuration puts a greater load on the network.
  - It is much more difficult to program and test software in n-tiered architectures than in two-tiered architectures, because more devices have to communicate



Client devices
Handle presentation logic

Web server
Handles Web-related business logic

Application server
Handles other business logic

Database server
Handles database-related tasks

Less Common Architectures

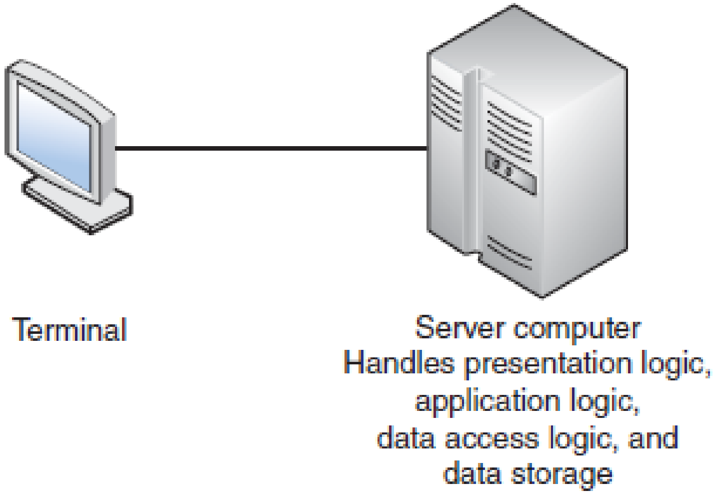
- The client–server architecture has become the predominant architecture in use today.
- Two other architectures are less commonly found, but still used in certain situations.
  - Server-Based Architectures
  - Client-Based Architectures

Server-Based Architectures

- The very first computing architectures were server based.
- the server (usually, a central mainframe computer) performing all four application functions.
- The clients (usually, terminals) enabled users to send and receive messages to and from the server computer.
    - The clients merely captured keystrokes and sent them to the server for processing, and accepted instructions from the server on what to display

Server-Based Architectures

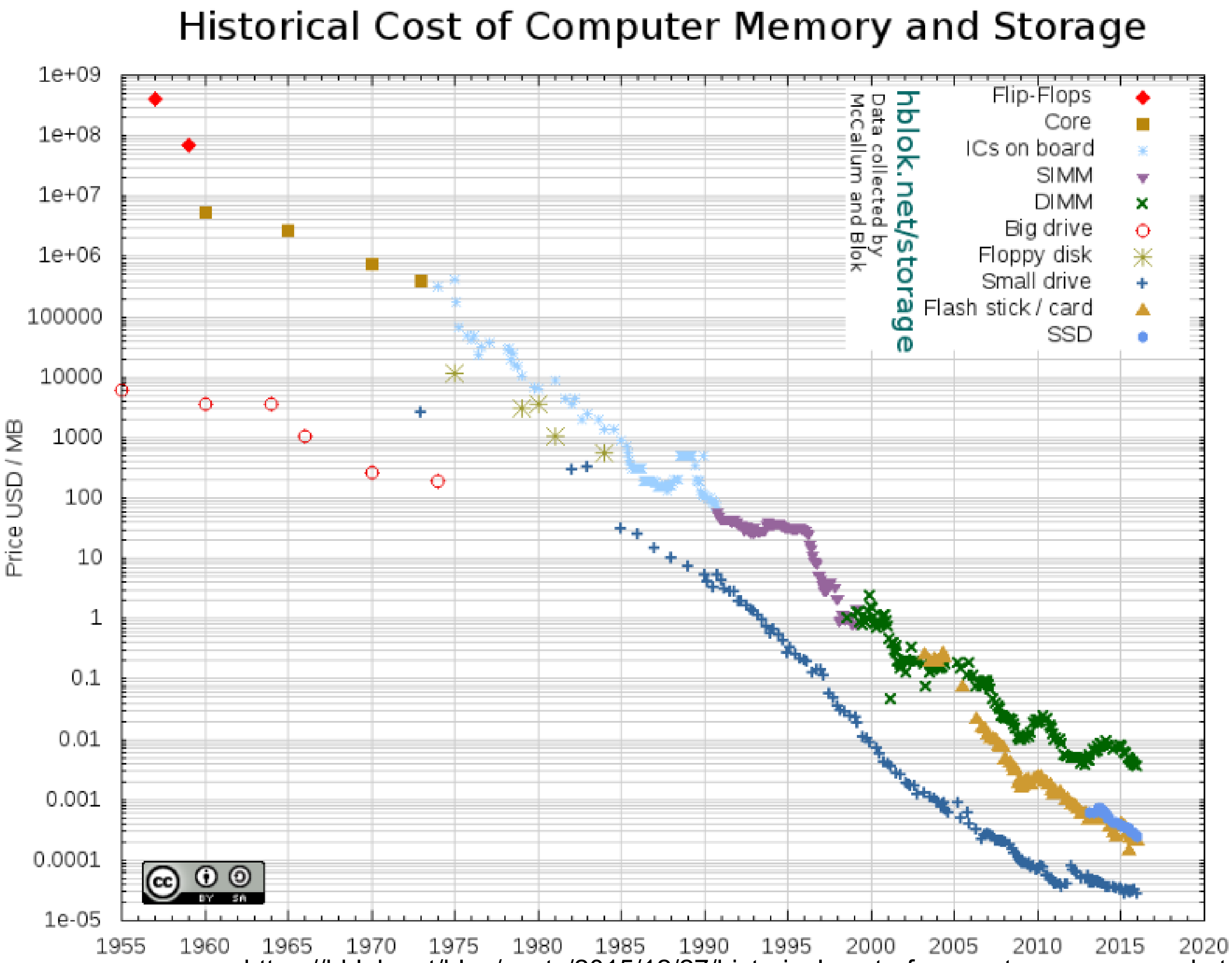
- Technical reasons.
- There is one point of control.
- Software development and software administration are simplified.
- Problems
  - demands for more and more applications
  - Growing number of users
  - Response time
  - Upgrade required a substantial financial commitment.
  - Increased capacity came only in large, expensive chunks.



Server-Based Architectures

EFOP-3.4.3-16-2016-00009

A felsőfokú oktatás minőségének és hozzáférhetőségének együttes javítása a Pannon Egyetemen



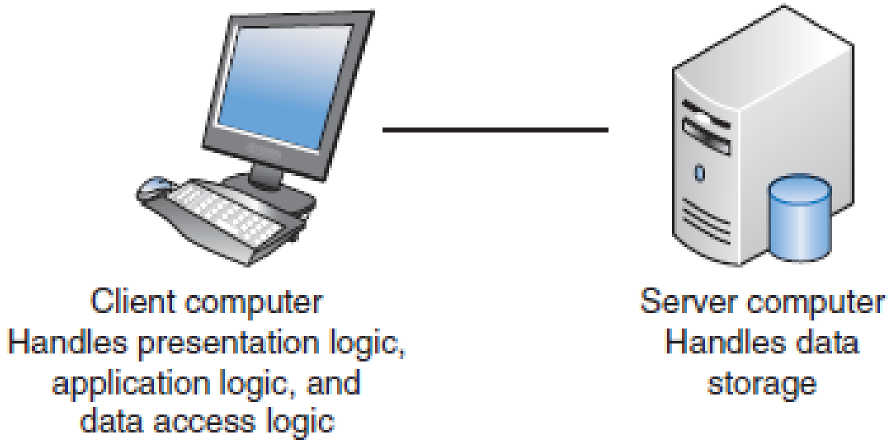
https://hblok.net/blog/posts/2015/12/27/historical-cost-of-computer-memory-and-storage-3

Server-Based Architectures

- Today, the server-based architecture remains a viable architecture choice
  - Zero client, or ultrathin client in a virtual desktop infrastructure (VDI).
  - A typical zero client device is a small box that connects a keyboard, mouse, monitor, and Ethernet connection to a remote server.
  - The server hosts everything: the client's operating system and all software applications.
  - Benefits: power usage, cost, vulnerability, security, non-business usage…

Client-Based Architectures

- The clients are microcomputers on a local area network.
- The server is a server computer on the same network.
- The application software on the client computers is responsible for the presentation logic, the application logic, and the data access logic.
- The server simply provides storage for the data.


Client computer
Handles presentation logic, application logic, and data access logic
Server computer
Handles data storage

Client-Based Architectures

- This simple architecture often works very well in situations
  - Low numbers of users
  - Limited data access requirements.
- Fundamental problems
  - All data on the server must travel to the client for processing.
  - In the client-based computing model, the data access logic is executed on the client system.
  - This can overload both the network and the power of the client computers.

Trends nowadays

- Advances in hardware, software, and networking have given rise to a number of new architecture options.
- Two advances that are currently getting a lot of attention:
  - Virtualization
  - Cloud Computing

Virtualization

- This term, refers to the creation of a virtual device or resource, such as a server or storage device.
- This term has become a common buzz word:
  - server virtualization
  - storage virtualization
  - network virtualization
- Server virtualization involves partitioning a physical server into smaller virtual servers.
- Software is used to divide the physical server into multiple virtual environments, called virtual or private servers.

Server virtualization

- Server virtualization involves partitioning a physical server into smaller virtual servers, called virtual or private servers.
- Today, a physical server device can be used to provide many virtual servers that are independent of each other, but co-reside on the same physical server.
    - Independent operating systems.
    - Each can be rebooted independently.
    - Less hardware is required.
    - Utilization can be optimized.
    - Operation costs

Server virtualization

- Storage virtualization involves combining multiple network storage devices into what appears to be single storage unit.
- A storage area network (SAN) uses storage virtualization to create a high-speed subnetwork of shared storage devices.
- In this environment, tasks such as backup, archiving, and recovery are easier and faster.

Cloud Computing

- The “cloud” in cloud computing can be defined as the set of
  - Hardware
  - Networks
  - Storage
  - Services
  - Interfaces

  that combine to deliver aspects of computing **as a service**.
- Based on user demand.
- Can be implemented in three ways
  - Private cloud
  - Public cloud.
  - Hybrid cloud.

Cloud Computing

- **Private clouds** offer activities and functions "as a service," but are deployed over a company intranet or hosted data centre.
- **Public clouds**, services are provided "as a service" over the Internet with little or no control over the underlying technology infrastructure.
- **Hybrid clouds** combine the power of both public and private clouds.

Cloud Computing

- Utilizing the cloud, the resources allocated can be increased or decreased based upon demand.
  - This elasticity, makes the cloud scalable—the cloud can scale up for periods of peak demand and scale down for times of less demand.
  - Cloud customers can obtain cloud resources in a straightforward fashion.
  - Cloud services typically have standardized APIs.
  - Dynamic pricing based on usage.
- Through the cloud computing model, the power of virtualization is converted into measurable business value.

CREATING AN ARCHITECTURE DESIGN

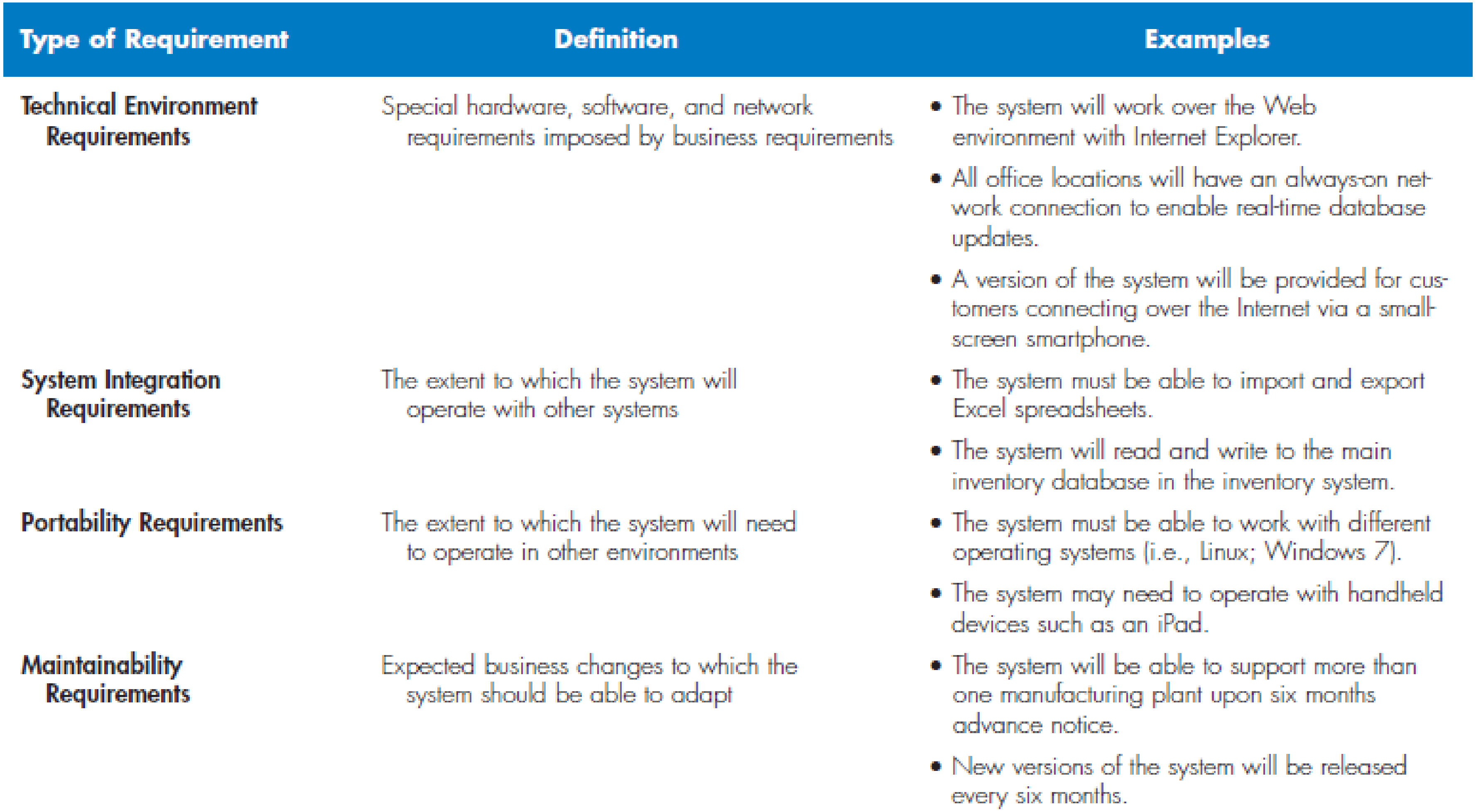
- Most systems are built to use the existing infrastructure.
  - The current infrastructure restricts the choice of architecture.
  - Corporate standards
  - Existing licensing agreements
  - Product/vendor relationships
- Creating an architecture design begins with the non-functional.

  requirements.
  - Refining the non-functional requirements.
  - Selecting the architecture to be used.
  - The non-functional requirements and the architecture design are used to develop the hardware and software specification.

CREATING AN ARCHITECTURE DESIGN

- There are four important primary types of non-functional requirements
  - Operational requirements
  - Performance requirements
  - Security requirements
  - Cultural and political requirements

Operational Requirements

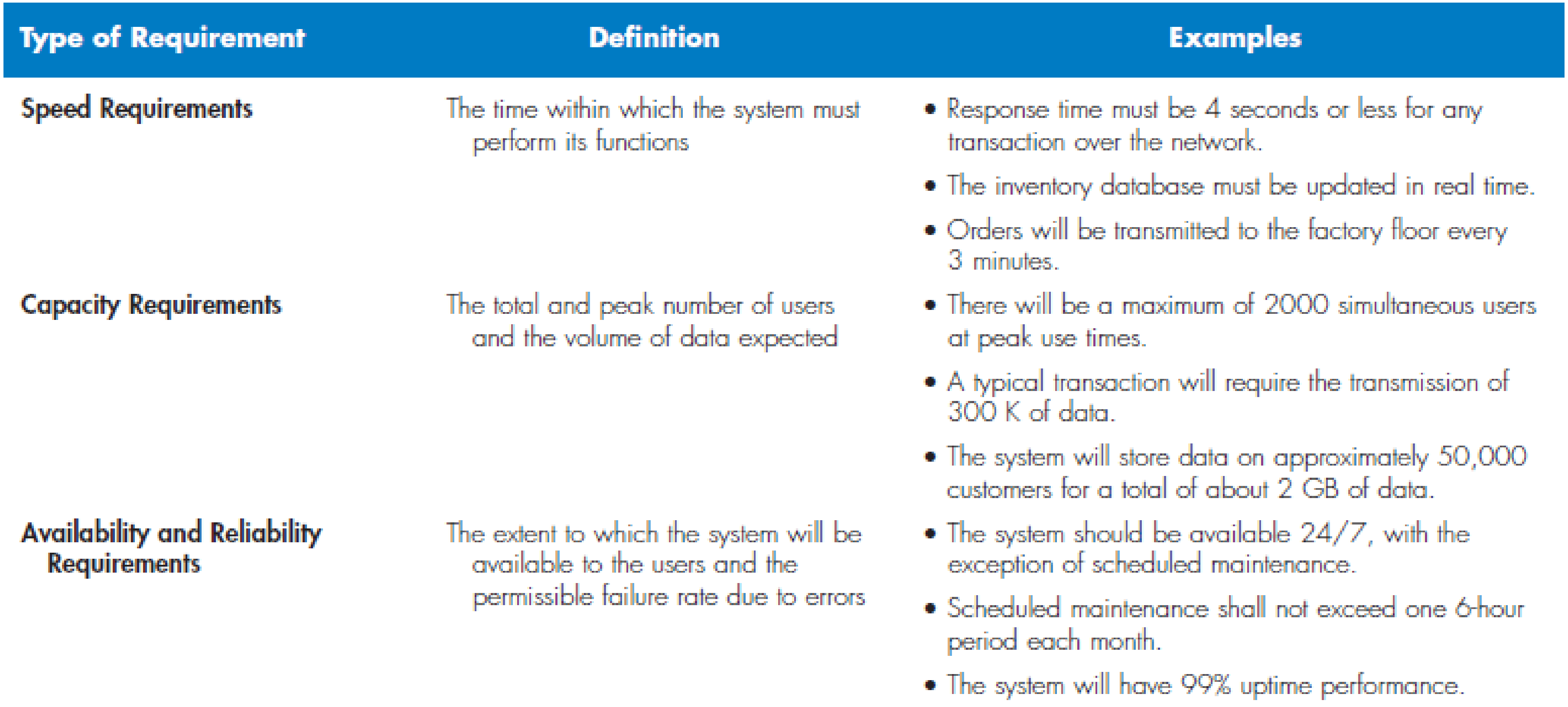


Type of Requirement	Definition	Examples
Technical Environment Requirements	Special hardware, software, and network requirements imposed by business requirements	• The system will work over the Web environment with Internet Explorer. • All office locations will have an always-on network connection to enable real-time database updates. • A version of the system will be provided for customers connecting over the Internet via a small-screen smartphone.
System Integration Requirements	The extent to which the system will operate with other systems	• The system must be able to import and export Excel spreadsheets. • The system will read and write to the main inventory database in the inventory system.
Portability Requirements	The extent to which the system will need to operate in other environments	• The system must be able to work with different operating systems (i.e., Linux; Windows 7). • The system may need to operate with handheld devices such as an iPad.
Maintainability Requirements	Expected business changes to which the system should be able to adapt	• The system will be able to support more than one manufacturing plant upon six months advance notice. • New versions of the system will be released every six months.

Performance Requirements

- It focuses on performance issues such as
  - Response time
  - Capacity
  - Reliability
- Each requirement must be measurable so that a benchmark comparison can be made.

Performance Requirements

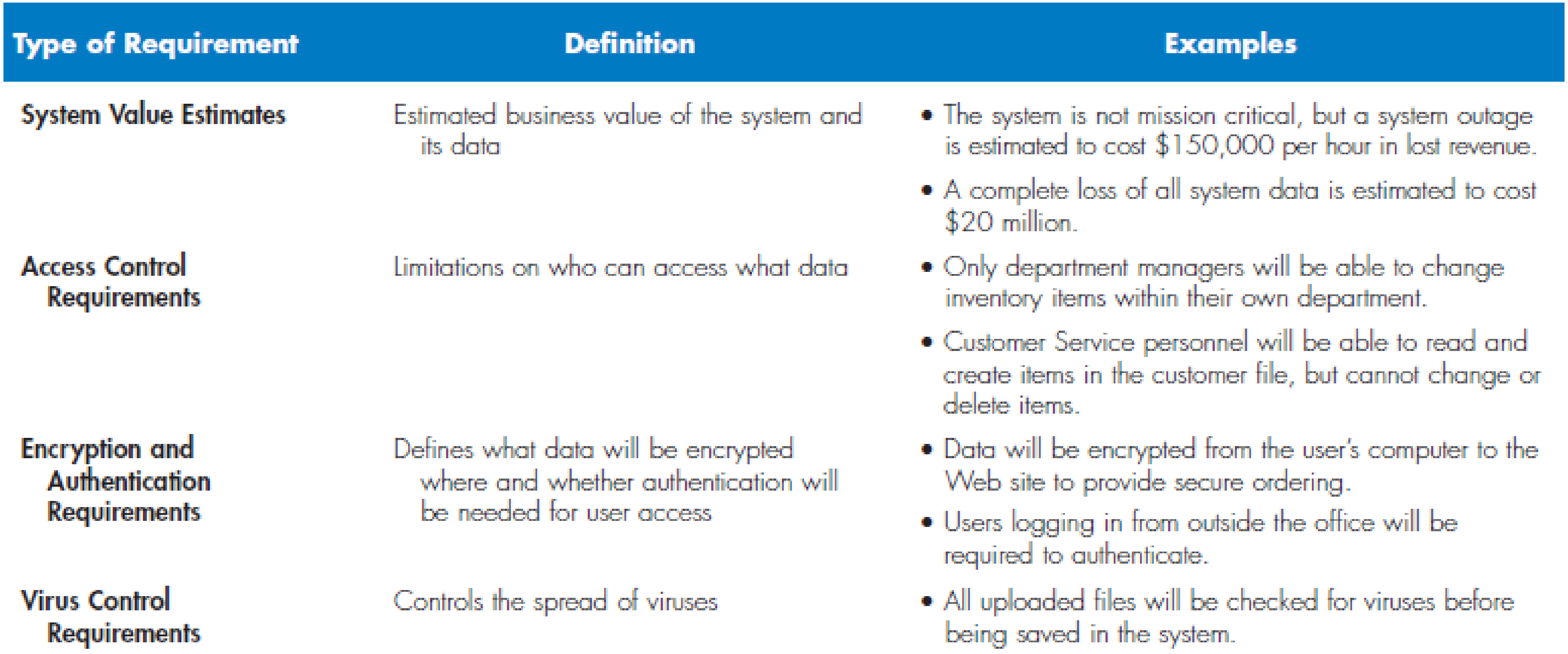


Type of Requirement	Definition	Examples
Speed Requirements	The time within which the system must perform its functions	• Response time must be 4 seconds or less for any transaction over the network. • The inventory database must be updated in real time. • Orders will be transmitted to the factory floor every 3 minutes.
Capacity Requirements	The total and peak number of users and the volume of data expected	• There will be a maximum of 2000 simultaneous users at peak use times. • A typical transaction will require the transmission of 300 K of data. • The system will store data on approximately 50,000 customers for a total of about 2 GB of data.
Availability and Reliability Requirements	The extent to which the system will be available to the users and the permissible failure rate due to errors	• The system should be available 24/7, with the exception of scheduled maintenance. • Scheduled maintenance shall not exceed one 6-hour period each month. • The system will have 99% uptime performance.

Security Requirements

- Security is the ability to protect the information system from disruption and data loss, whether caused by
  - An intentional act (e.g., a hacker or a terrorist attack)
  - A random event (e.g., disk failure, tornado).
- Security is primarily the responsibility of the operations group—the staff responsible for installing and operating security controls such as firewalls, intrusion detection systems, and routine backup and recovery operations.

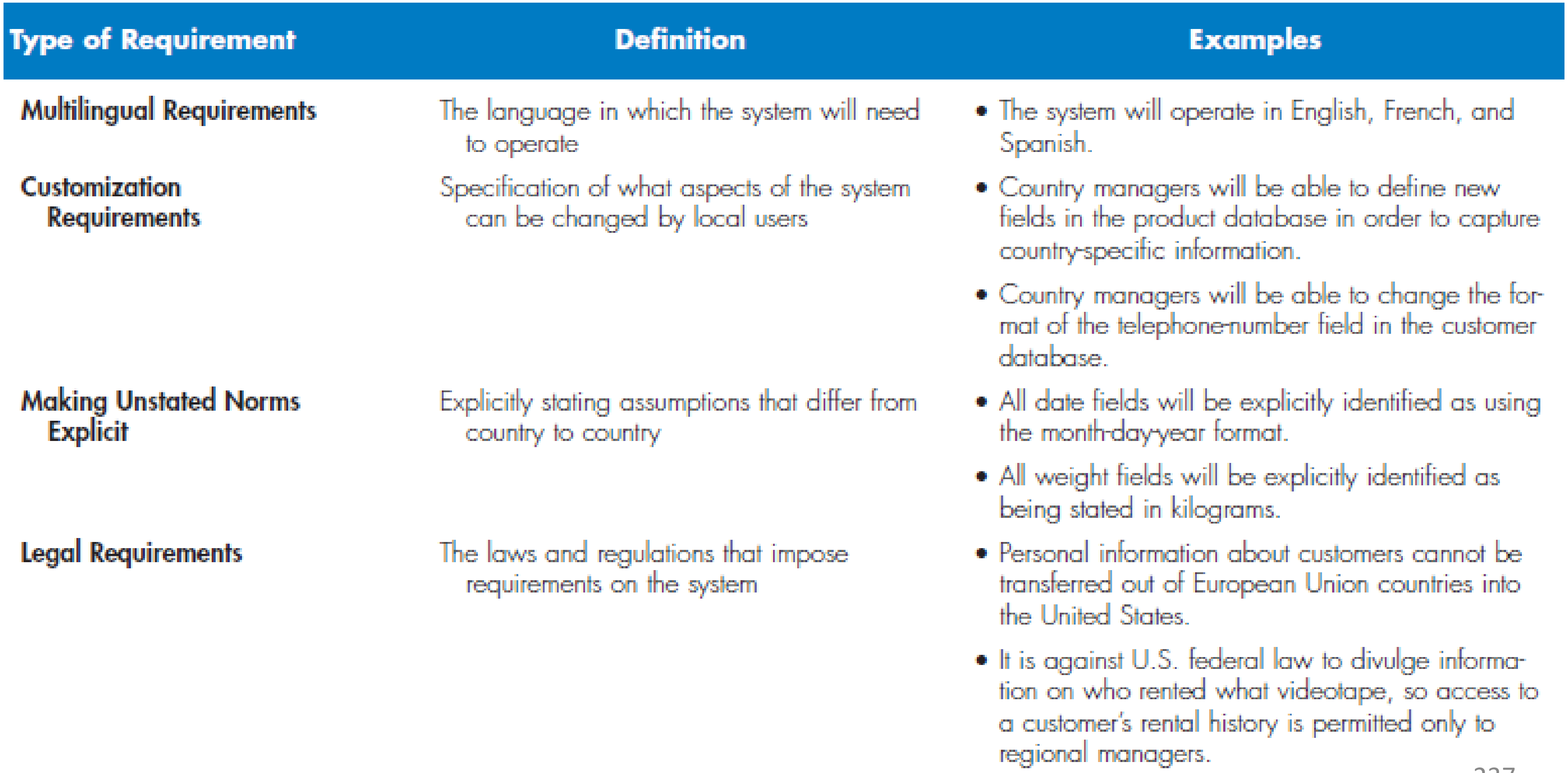
Security Requirements



Type of Requirement	Definition	Examples
System Value Estimates	Estimated business value of the system and its data	• The system is not mission critical, but a system outage is estimated to cost $150,000 per hour in lost revenue. • A complete loss of all system data is estimated to cost $20 million.
Access Control Requirements	Limitations on who can access what data	• Only department managers will be able to change inventory items within their own department. • Customer Service personnel will be able to read and create items in the customer file, but cannot change or delete items.
Encryption and Authentication Requirements	Defines what data will be encrypted where and whether authentication will be needed for user access	• Data will be encrypted from the user's computer to the Web site to provide secure ordering. • Users logging in from outside the office will be required to authenticate.
Virus Control Requirements	Controls the spread of viruses	• All uploaded files will be checked for viruses before being saved in the system.

Cultural and Political Requirements

- Cultural and political requirements are specific to the countries in which the system will be used.



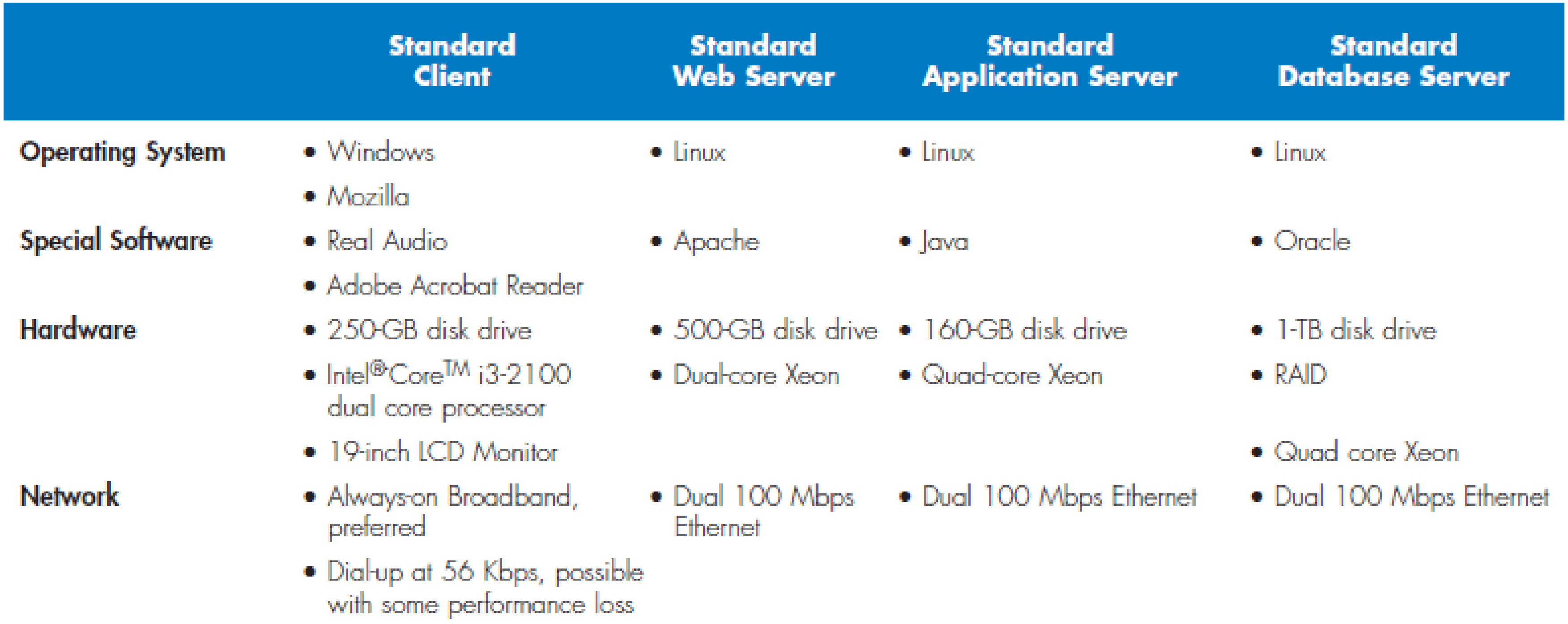
Type of Requirement	Definition	Examples
Multilingual Requirements	The language in which the system will need to operate	• The system will operate in English, French, and Spanish.
Customization Requirements	Specification of what aspects of the system can be changed by local users	• Country managers will be able to define new fields in the product database in order to capture country-specific information. • Country managers will be able to change the format of the telephone-number field in the customer database.
Making Unstated Norms Explicit	Explicitly stating assumptions that differ from country to country	• All date fields will be explicitly identified as using the month-day-year format. • All weight fields will be explicitly identified as being stated in kilograms.
Legal Requirements	The laws and regulations that impose requirements on the system	• Personal information about customers cannot be transferred out of European Union countries into the United States. • It is against U.S. federal law to divulge information on who rented what videotape, so access to a customer's rental history is permitted only to regional managers.

Designing the Architecture

- Business requirements dominate other considerations.
  - It is still important to work through and refine the remaining non-functional requirements.
- If the technical environment requirements do not specify a specific architecture, then the other non-functional requirements become important.
- Hardware and software specification.
  - It is a document that describes what hardware and software are needed to support the application.

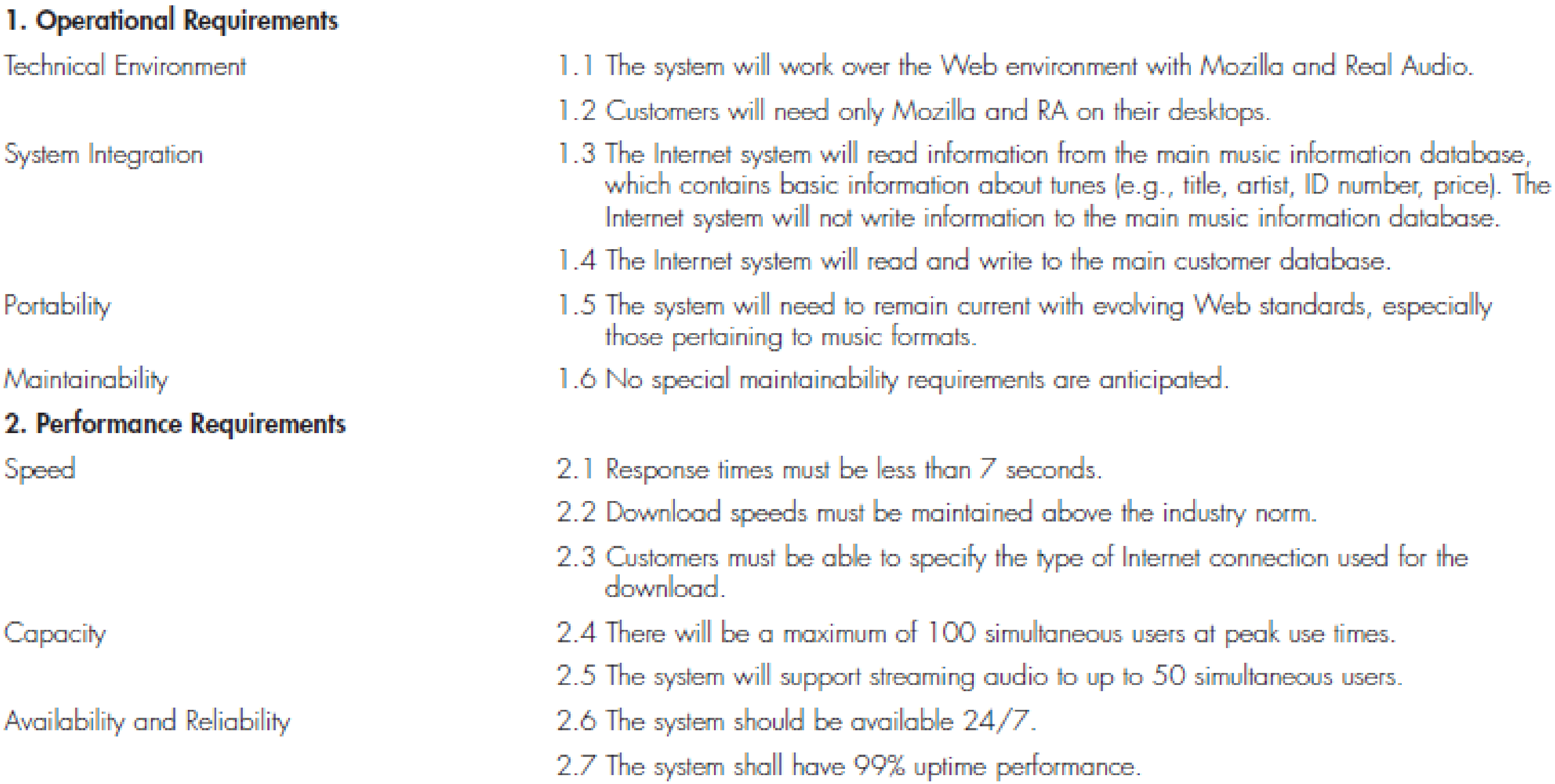
Designing the Architecture

- Example for Tune Source's Digital Music Download system



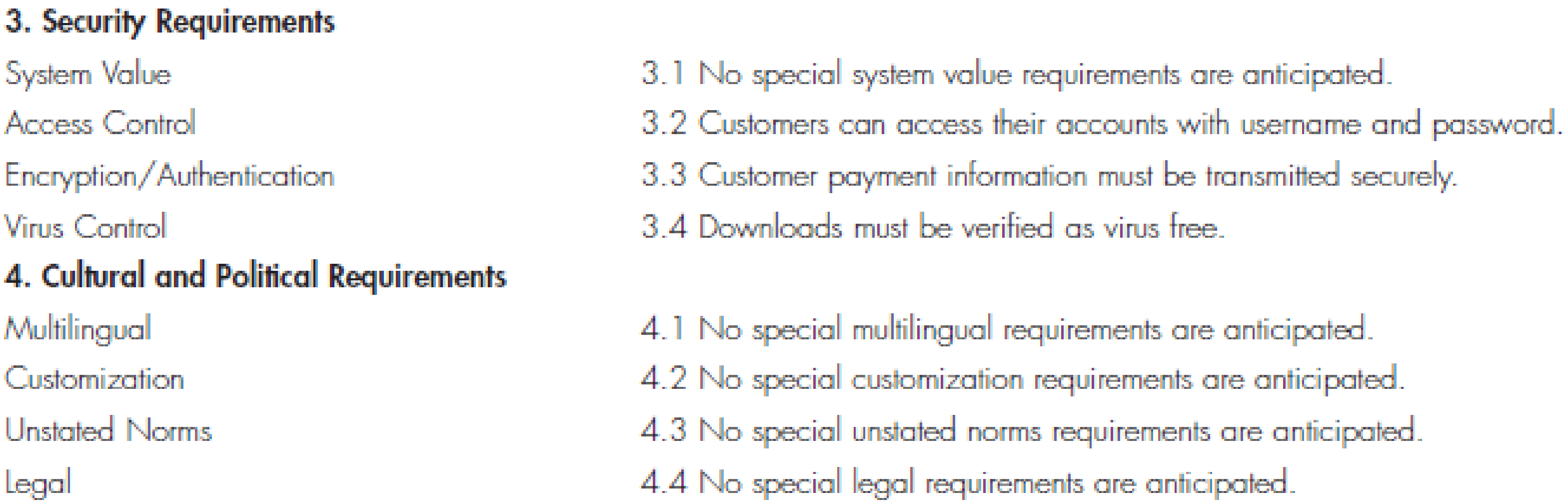
	Standard Client	Standard Web Server	Standard Application Server	Standard Database Server
Operating System	• Windows • Mozilla	• Linux	• Linux	• Linux
Special Software	• Real Audio • Adobe Acrobat Reader	• Apache	• Java	• Oracle
Hardware	• 250-GB disk drive • Intel®Core^{TM} i3-2100 dual core processor • 19-inch LCD Monitor	• 500-GB disk drive • Dual-core Xeon	• 160-GB disk drive • Quad-core Xeon	• 1-TB disk drive • RAID • Quad core Xeon
Network	• Always-on Broadband, preferred • Dial-up at 56 Kbps, possible with some performance loss	• Dual 100 Mbps Ethernet	• Dual 100 Mbps Ethernet	• Dual 100 Mbps Ethernet

Designing the Architecture



1. Operational Requirements	
Technical Environment	1.1 The system will work over the Web environment with Mozilla and Real Audio.
	1.2 Customers will need only Mozilla and RA on their desktops.
System Integration	1.3 The Internet system will read information from the main music information database, which contains basic information about tunes (e.g., title, artist, ID number, price). The Internet system will not write information to the main music information database.
	1.4 The Internet system will read and write to the main customer database.
Portability	1.5 The system will need to remain current with evolving Web standards, especially those pertaining to music formats.
Maintainability	1.6 No special maintainability requirements are anticipated.
2. Performance Requirements	
Speed	2.1 Response times must be less than 7 seconds.
	2.2 Download speeds must be maintained above the industry norm.
	2.3 Customers must be able to specify the type of Internet connection used for the download.
Capacity	2.4 There will be a maximum of 100 simultaneous users at peak use times.
	2.5 The system will support streaming audio to up to 50 simultaneous users.
Availability and Reliability	2.6 The system should be available 24/7.
	2.7 The system shall have 99% uptime performance.

Designing the Architecture



3. Security Requirements	
System Value	3.1 No special system value requirements are anticipated.
Access Control	3.2 Customers can access their accounts with username and password.
Encryption/Authentication	3.3 Customer payment information must be transmitted securely.
Virus Control	3.4 Downloads must be verified as virus free.
4. Cultural and Political Requirements	
Multilingual	4.1 No special multilingual requirements are anticipated.
Customization	4.2 No special customization requirements are anticipated.
Unstated Norms	4.3 No special unstated norms requirements are anticipated.
Legal	4.4 No special legal requirements are anticipated.

Designing the Architecture

- The architecture group and the Internet project team decided that the only components that needed to be acquired for the project were
  - a database server
  - a Web server
  - and five new client computers for the marketing group who will create the promotional campaigns.
- They developed a hardware and software specification for these components, prepared an RFP, and worked with the purchasing department to acquire the hardware and software.

THANKS FOR ATTENTION!

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

EFOP-3.4.3-16-2016-00009

A felsőfokú oktatás minőségének és hozzáférhetőségének együttes javítása a Pannon Egyetemen

SYSTEM ANALYSIS AND SYSTEM DESIGN

8. INTERFACE DESIGN

Imre Budai

SZÉCHENYI 2020

Magyarország Kormánya

Európai Unió
Európai Szociális
Alap

BEFEKTETÉS A JÖVŐBE

Objectives

- Describe several fundamental user interface design principles.
- Explain the process of user interface design.
- Discuss how to design the user interface structure.
- Explain how to design the user interface standards.
- Be able to design a user interface.

Introduction

- It defines how the system will interact with external entities.
  - User interface

    Defines the way in which the users will interact with the system and the nature of the inputs and outputs that the system accepts and produces.
  - System interface

    Exchanges information with other systems.

- We focus on the design of user interfaces.

Introduction

- The user interface includes three fundamental parts.
  - The navigation mechanism

    The way in which the user gives instructions to the system and tells it what to do.
  - The input mechanism

    The way in which the system captures information.
  - The output mechanism

    The way in which the system provides information to the user or to other systems.
- Each of these is conceptually different, but all are closely intertwined.

Introduction

- The study of human-computer interaction (HCI) focuses on improving the interactions between users and computers by making computers more usable and receptive to the user’s needs.

- Each of these is conceptually different, but all are closely intertwined.

- Fundamental user interface design principles.
- Overview of the user interface design process.

Introduction

- Focus on
  - Design of Web-based interfaces.
  - Graphical user interfaces

    That use windows, menus, icons, and a mouse.

- Although text-based interfaces are still commonly used on mainframes and UNIX systems.

Principles

- Main challenge: Using space effectively.

  Simply put, there is more information to present than room to present it.

  - Balance the need for simplicity and pleasant appearance against the need to present the information across multiple pages or screens, which decreases simplicity.
- Fundamental interface design principles
  - Layout
  - Content awareness
  - Aesthetics
  - User experience
  - Consistency
  - Minimize user effort

Layout

- Layout of the screen, form, or report.
- Organizing areas of the screen or document for different purposes and using those areas consistently throughout the user interface.
- Most software designed for personal computers.
  - „Standard” Windows or Macintosh approach for screen layout.
  - Three main areas: top area, middle area , bottom area
- Web based applications.
  - Multiple layout areas are used.
  - Each is organized to provide different functions and navigation within different parts of the system

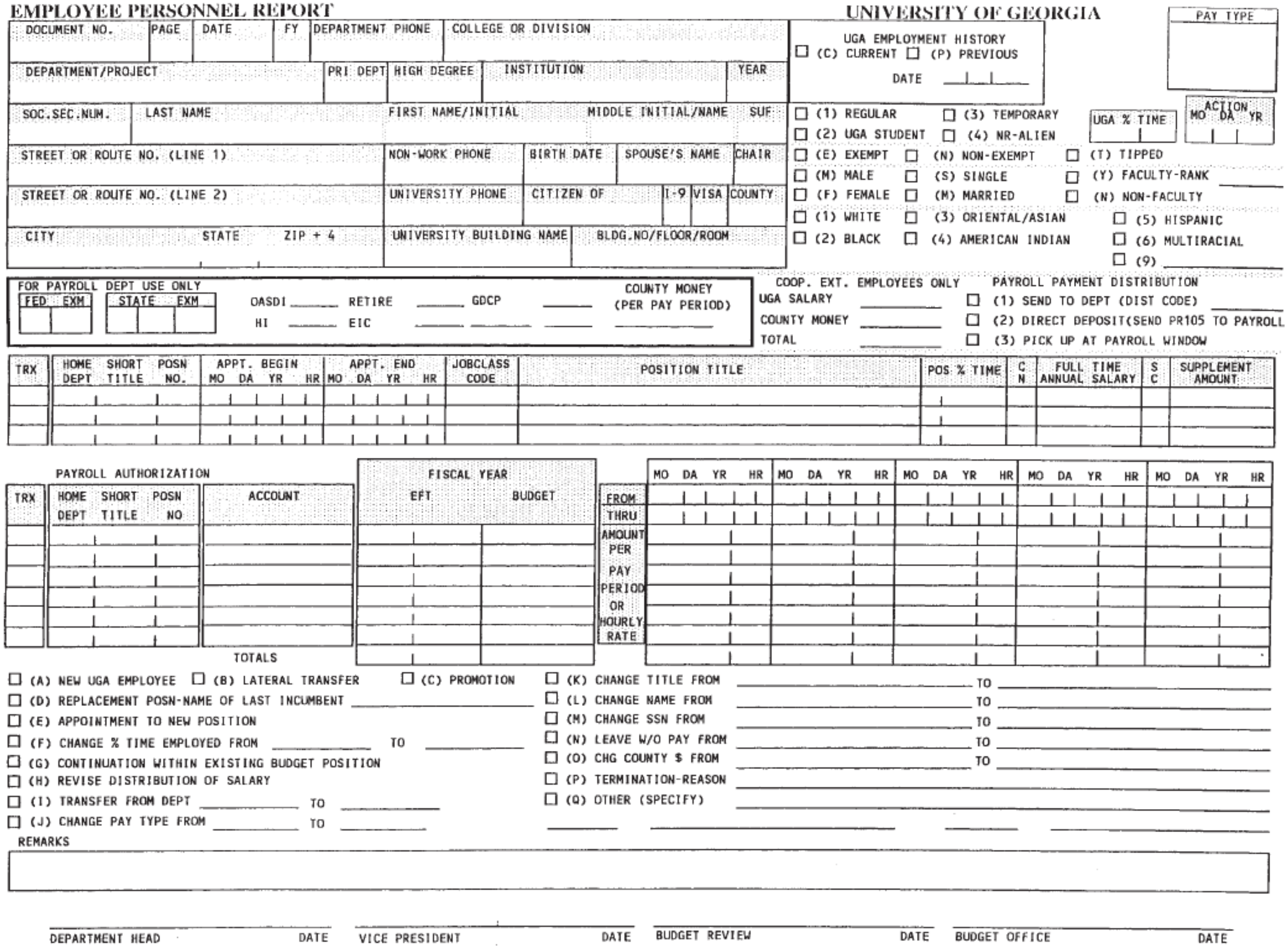
Content Awareness

- The ability of an interface to make the user aware of the information it contains.
  - With the least amount of effort by the user.
- It is a general requirement, however it is particularly important for forms or reports that are used quickly or irregularly.
- All interfaces should have
  - Titles
  - Menus
    - Where the user is.
    - Where the user came from.

Aesthetics

- Designing interfaces that are pleasing to the eye.
- Interfaces do not have to be works of art, but they do need to be functional and inviting to use.
- Temptation to squeeze as much information as possible.
- But, in most cases **less is more**…

Aesthetics



EMPLOYEE PERSONNEL REPORT

UNIVERSITY OF GEORGIA

PAY TYPE

DOCUMENT NO. | PAGE | DATE | FY | DEPARTMENT PHONE | COLLEGE OR DIVISION

UGA EMPLOYMENT HISTORY
□ (C) CURRENT □ (P) PREVIOUS
DATE

DEPARTMENT/PROJECT | PRI DEPT | HIGH DEGREE | INSTITUTION | YEAR

SOC.SEC.NUM. | LAST NAME | FIRST NAME/INITIAL | MIDDLE INITIAL/NAME | SUF

STREET OR ROUTE NO. (LINE 1) | NON-WORK PHONE | BIRTH DATE | SPOUSE'S NAME | CHAIR

STREET OR ROUTE NO. (LINE 2) | UNIVERSITY PHONE | CITIZEN OF | I-9 | VISA | COUNTY

CITY | STATE | ZIP + 4 | UNIVERSITY BUILDING NAME | BLDG.NO/FLOOR/ROOM

□ (1) REGULAR □ (3) TEMPORARY
□ (2) UGA STUDENT □ (4) NR-ALIEN
UGA % TIME
ACTION MO DA YR
□ (E) EXEMPT □ (N) NON-EXEMPT □ (T) TIPPED
□ (M) MALE □ (S) SINGLE □ (Y) FACULTY-RANK
□ (F) FEMALE □ (M) MARRIED □ (N) NON-FACULTY
□ (1) WHITE □ (3) ORIENTAL/ASIAN □ (5) HISPANIC
□ (2) BLACK □ (4) AMERICAN INDIAN □ (6) MULTIRACIAL
□ (9)

FOR PAYROLL DEPT USE ONLY
FED EXM | STATE EXM
OASDI ______ RETIRE ______ GDCP ______
HI ______ EIC ______
COUNTY MONEY (PER PAY PERIOD)

COOP. EXT. EMPLOYEES ONLY
UGA SALARY
COUNTY MONEY
TOTAL

PAYROLL PAYMENT DISTRIBUTION
□ (1) SEND TO DEPT (DIST CODE)
□ (2) DIRECT DEPOSIT(SEND PR105 TO PAYROLL
□ (3) PICK UP AT PAYROLL WINDOW

TRX	HOME DEPT	SHORT TITLE	POSN NO.	APPT. BEGIN MO DA YR HR	APPT. END MO DA YR HR	JOBCLASS CODE	POSITION TITLE	POS % TIME	C N	FULL TIME ANNUAL SALARY	S C	SUPPLEMENT AMOUNT

PAYROLL AUTHORIZATION

TRX	HOME DEPT	SHORT TITLE	POSN NO	ACCOUNT	FISCAL YEAR EFT	FISCAL YEAR BUDGET
				TOTALS		

	MO DA YR HR	MO DA YR HR	MO DA YR HR	MO DA YR HR	MO DA YR HR
FROM					
THRU					
AMOUNT PER PAY PERIOD OR HOURLY RATE					

□ (A) NEW UGA EMPLOYEE □ (B) LATERAL TRANSFER □ (C) PROMOTION
□ (D) REPLACEMENT POSN-NAME OF LAST INCUMBENT ______
□ (E) APPOINTMENT TO NEW POSITION
□ (F) CHANGE % TIME EMPLOYED FROM ______ TO ______
□ (G) CONTINUATION WITHIN EXISTING BUDGET POSITION
□ (H) REVISE DISTRIBUTION OF SALARY
□ (I) TRANSFER FROM DEPT ______ TO ______
□ (J) CHANGE PAY TYPE FROM ______ TO ______

□ (K) CHANGE TITLE FROM ______ TO ______
□ (L) CHANGE NAME FROM ______ TO ______
□ (M) CHANGE SSN FROM ______ TO ______
□ (N) LEAVE W/O PAY FROM ______ TO ______
□ (O) CHG COUNTY $ FROM ______ TO ______
□ (P) TERMINATION-REASON ______
□ (Q) OTHER (SPECIFY) ______

REMARKS

DEPARTMENT HEAD DATE VICE PRESIDENT DATE BUDGET REVIEW DATE BUDGET OFFICE DATE

Aesthetics

- Preferences in general
  - Novice or infrequent users: 50% or less density
  - Expert users: high density (90%)
- Text design is also important – depending on media type.
  - Font size
  - Font type
  - Emphasis
  - Colour

User Experience

- Designing the user interface with the users' level of computer experience in mind.
- The user interface should be designed for both types:
  - Experienced users.
  - Novice users.
- Ease of learning.
- Ease of use.

Often, these two objectives are complementary.

Consistency

- Consistency in design is probably the single most important factor in making a system simple to use.
- It enables users to predict what will happen.

  When interfaces are consistent, users can interact with one part of the system and then know how to interact with the rest
- Consistency occurs at many different levels.
  - In the navigation controls - using the same icon or command
  - Consistency in terminology
  - Consistency in report and form design

Minimize User Effort

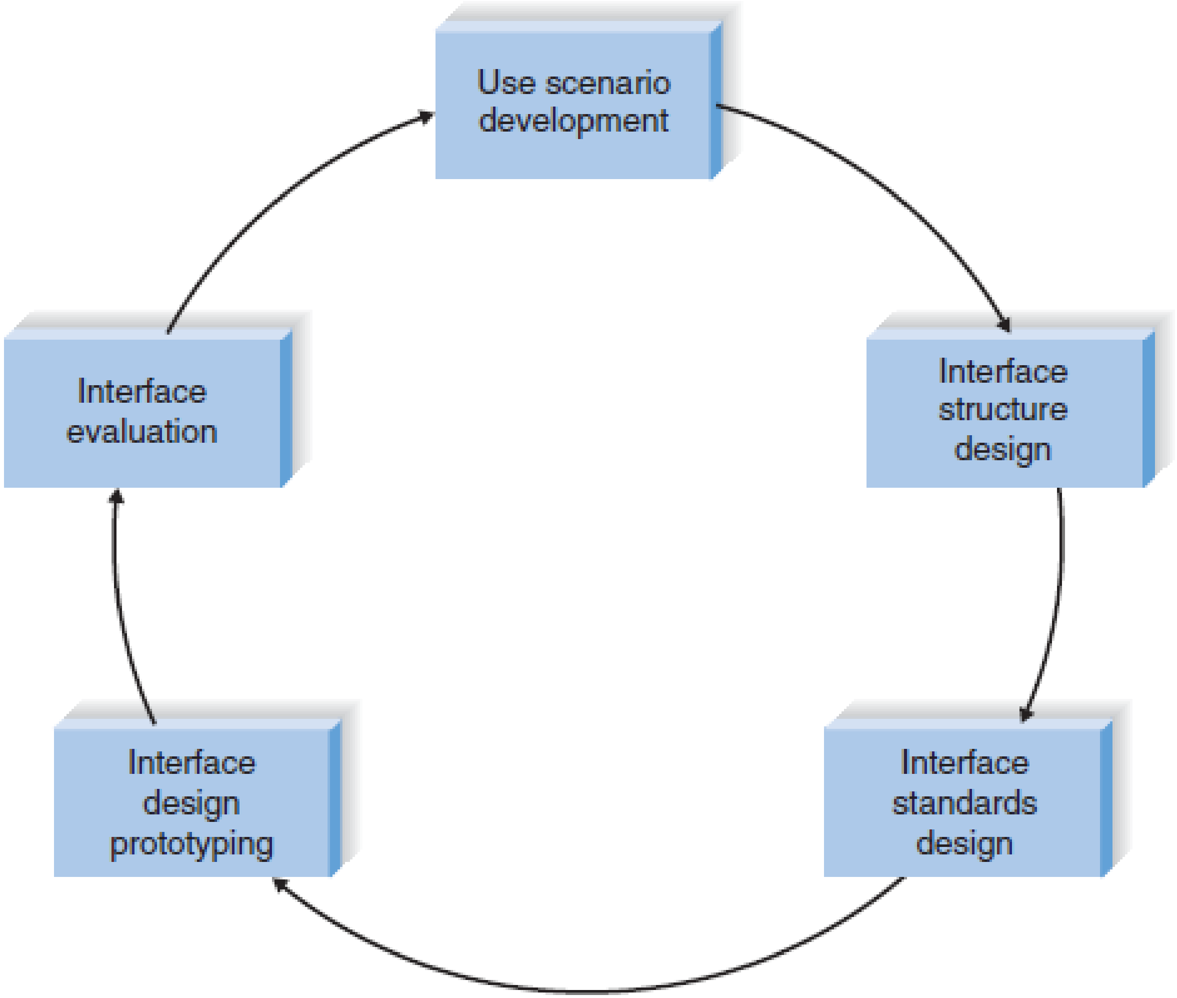
- It means using the fewest possible mouse clicks or keystrokes to move from one part of the system to another.
- Three-clicks rule:

  Users should be able to go from the start or main menu of a system to the information or action they want in no more than three mouse clicks or three keystrokes.

User Interface Design Process

- It is a five-step process that is iterative.
    - Examining the DFDs, use cases and interview users – use scenarios.
    - Developing the interface structure diagram (ISD) – the basic structure of the interface
    - Designing interface standards - the basic design elements.
    - Creating interface design prototypes - navigation controls, input/output screens, forms and reports.
    - Interface evaluation - improvement.

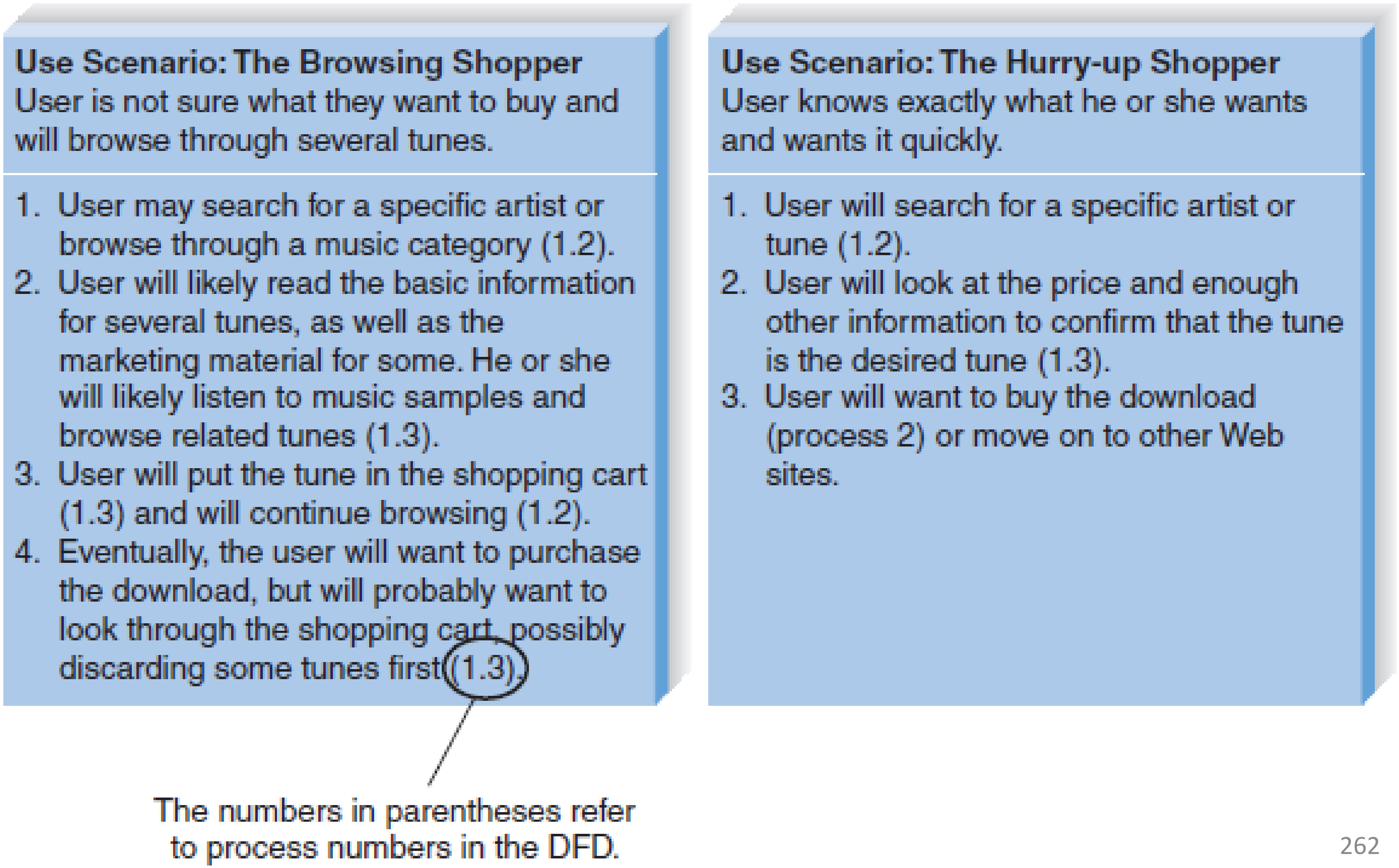
User Interface Design Process


Use scenario development
Interface structure design
Interface standards design
Interface design prototyping
Interface evaluation

Use Scenario Development

- A use scenario is an outline of the steps that the users perform to accomplish some part of their work.
- It is one commonly used path through a use case.
- The DFD was designed to model all possible uses of the system—that is, its complete functionality or all possible paths through the use case.
  - But use scenarios are just one path through the use case.
- Use scenarios are presented in a simple narrative description that is tied to the DFD.

Use Scenario Development



Use Scenario: The Browsing Shopper
User is not sure what they want to buy and will browse through several tunes.

1. User may search for a specific artist or browse through a music category (1.2).
2. User will likely read the basic information for several tunes, as well as the marketing material for some. He or she will likely listen to music samples and browse related tunes (1.3).
3. User will put the tune in the shopping cart (1.3) and will continue browsing (1.2).
4. Eventually, the user will want to purchase the download, but will probably want to look through the shopping cart, possibly discarding some tunes first (1.3).

Use Scenario: The Hurry-up Shopper
User knows exactly what he or she wants and wants it quickly.

1. User will search for a specific artist or tune (1.2).
2. User will look at the price and enough other information to confirm that the tune is the desired tune (1.3).
3. User will want to buy the download (process 2) or move on to other Web sites.

The numbers in parentheses refer to process numbers in the DFD.

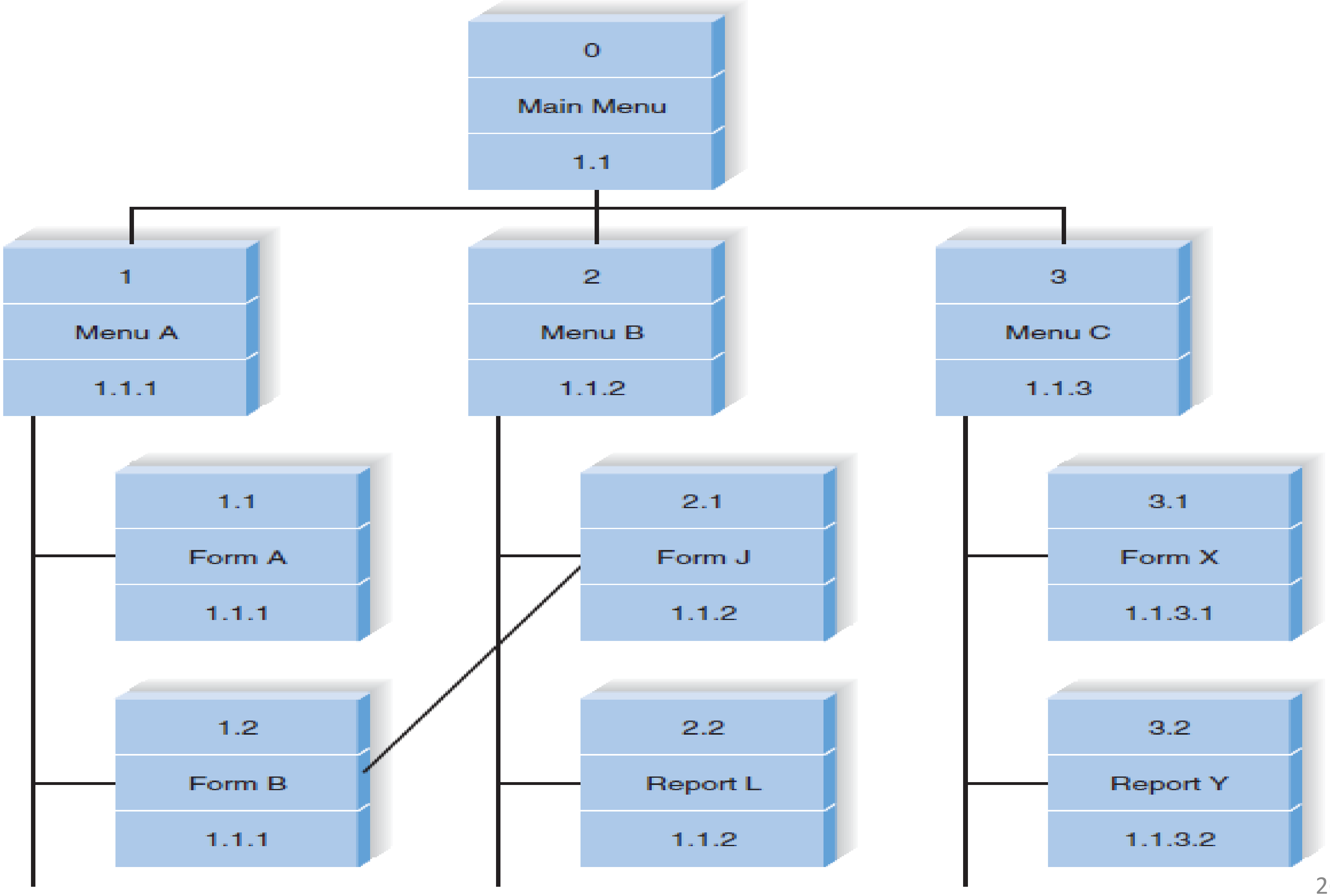
Use Scenario Development

- The key point in using use scenarios for interface design is **not** to document **all possible** use scenarios **but** to **describe the handful of most commonly occurring** use scenarios so that the interface can be designed to **enable the most common uses** to be performed **simply** and **easily**.

Interface Structure Design

- It defines the basic components of the interface and how they work together to provide functionality to users.
- It is used to show how all the screens, forms, and reports used by the system are related and how the user moves from one to another.
- Each interface element on an ISD is a box with:
  - A unique number (at the top)
  - A unique name (in the middle)
  - The DFD process that is supported by the interface (at the bottom)

Interface Structure Design

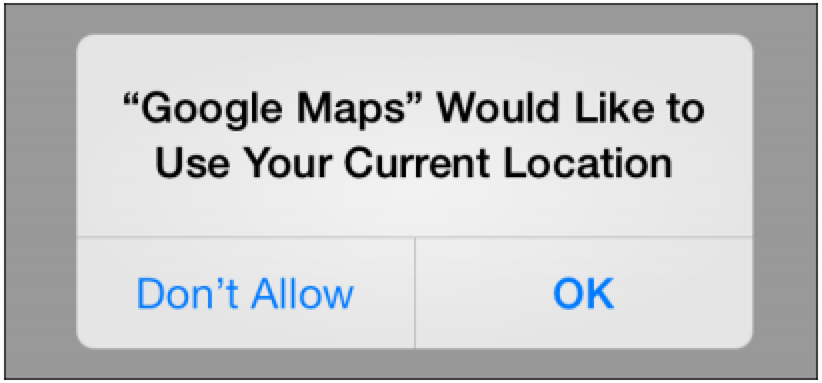


Interface Standards Design

- The interface standards are the basic design elements that are common across the individual screens, forms, and reports within the system.
  - Interface Metaphor
  - Interface Templates
  - Interface Objects
  - Interface Actions
  - Interface Icons

Interface metaphor

- It defines how the interface will work.
  - Explicit.
  - Implicit.
  - In some cases, the metaphor is so obvious that it requires no thought.


“Google Maps” Would Like to Use Your Current Location

or „**Trashcan**”

Interface templates

- It defines the general appearance of all screens in the information system.
    - Basic layout.
    - Colour schemes.
    - Replacement or cascading.
    - Placement and order for common interface actions.
- It defines whether windows will replace one another on the screen or will cascade on top of each other.
- The template draws together the other major interface design elements: metaphors, objects, actions, and icons.

Interface objects

- The fundamental building blocks of the system such as the entities and data stores.
- In many cases, the object names are straightforward, such as calling the shopping cart the “shopping cart.”
- In other cases, it is not simple.
    - For e.g.: Tune Source has chosen to call its digital music downloads “tunes.” Some people may refer to individual music selections as “tracks” or “cuts.”
- The object names should be easily understood and should help promote the interface metaphor.

Interface actions

- It specifies
  - Navigation and command language style (menus).
  - Grammar (object–action order – later: “Navigation Design”).
- The template gives names to the most commonly used interface actions in the navigation design.
  - For e.g., “**buy**” versus “**purchase**,” or “**exit**” versus “**quit**”.

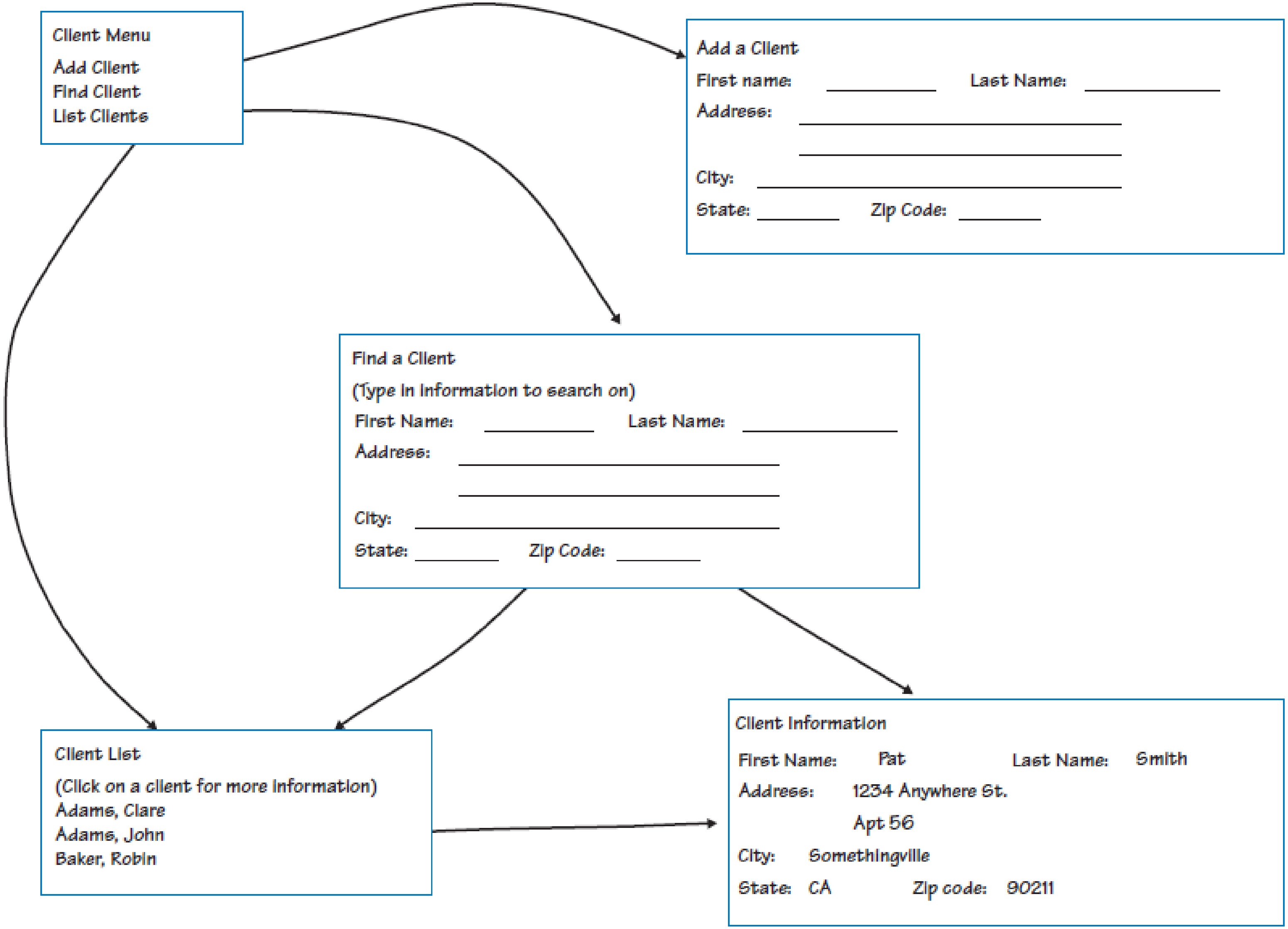
Interface icons

- Icons are pictures that will appear on command buttons as well as in reports and forms to highlight important information.
- The simplest and best approach is to adopt well known/understood icons.
- The use of icons can sometimes cause more confusion than insight.
- Using text tool tips.

Interface Design Prototyping

- It is a mock-up or a simulation of a computer screen, form, or report.
- It is prepared for each interface in the system to show the users and the programmers how the system will perform.
  - Storyboards
  - HTML prototypes
  - Language prototypes

Storyboards



Client Menu

Add Client
Find Client
List Clients

Add a Client

First name: ________ Last Name: __________

Address: ________________________

____

City: ___________________________

State: ______ Zip Code: ______

Find a Client

(Type in information to search on)

First Name: ________ Last Name: __________

Address: ________________________

____

City: ___________________________

State: ______ Zip Code: ______

Client List

(Click on a client for more information)
Adams, Clare
Adams, John
Baker, Robin

Client Information

First Name: Pat Last Name: Smith

Address: 1234 Anywhere St.
Apt 56

City: Somethingville

State: CA Zip code: 90211

HTML prototypes

- One of the most common types of interface design prototypes used today.
- The users can interact with the pages by clicking on buttons and entering pretend data into forms.
- They enable users to interact with the system – better sense of how to navigate among the different screens.

Language Prototype

- It is built in the actual language or by the actual tool that will be used to build the system.
- It takes longer to develop than other techniques.
- They have the distinct advantage of showing exactly what the screens will look like.
  - The user does not have to guess about the shape or position of the elements on the screen.

Interface Evaluation

- How to improve the interface design.
- Interface design is subjective – differences in personal preferences
- It is to have as many people as possible evaluate the interface.
- Ideally, interface evaluation should be performed while the system is being designed.
  - Any major design problems can be identified and corrected.
  - Save time and cost of programming have been spent on a weak design.

Interface Evaluation

- Four common approaches for evaluation:
  - Heuristic evaluation
  - Walk-through evaluation
  - Interactive evaluation
  - Formal usability testing
- The different parts of a system can be evaluated by different techniques.

Heuristic evaluation

- The weakest type of evaluation.
- A heuristic evaluation examines the interface by comparing it to a set of heuristics, or principles, for interface design.
- There is a checklist of interface design principles.
- This technique does not involve the users.

Walk-through evaluation

- It is a meeting conducted with the users who will ultimately have to operate the system.
- The project team shows the prototypes.

  The storyboard or actually demonstrates the HTML or language prototype and explains how the interface will be used.
- The users identify improvements to each of the interfaces that are presented.

Interactive evaluation

- Used with HTML or language prototype.
- One-on-one sessions with the users and the members of the project team.
  - User tells what he or she likes and doesn’t like and what additional information or functionality is needed.
  - Team members record the situation when the user appears to be unsure of what to do, makes mistakes, or misinterprets the meaning of an interface component.
  - If those recurs across several evaluation sessions with several users then interface improvement is need.

Formal usability testing

- It is a very formal process – and very expensive
- It can be used only with language prototypes (and systems that have been completely built and are awaiting installation or shipping).
- Done in a special lab equipped with video cameras and special software.
  - The user is given a specific set of use scenarios.
  - Some initial instructions.
  - Without any help by project team member.

Navigation design

- It enables the user to
  - Enter commands to navigate through the system
  - Perform actions to
    - Enter information
    - Review information

    it contains.
- The goal is to make the system as simple as possible to use.
- A good navigation component is one that the user never really notices.

Navigation design

Basic Principles

- All controls should be
    - Clear
    - Understandable
    - placed in an intuitive location on the screen.
- Prevent Mistakes
- Simplify Recovery from Mistakes
- Use Consistent Grammar Order

Navigation design

Types of Navigation Controls

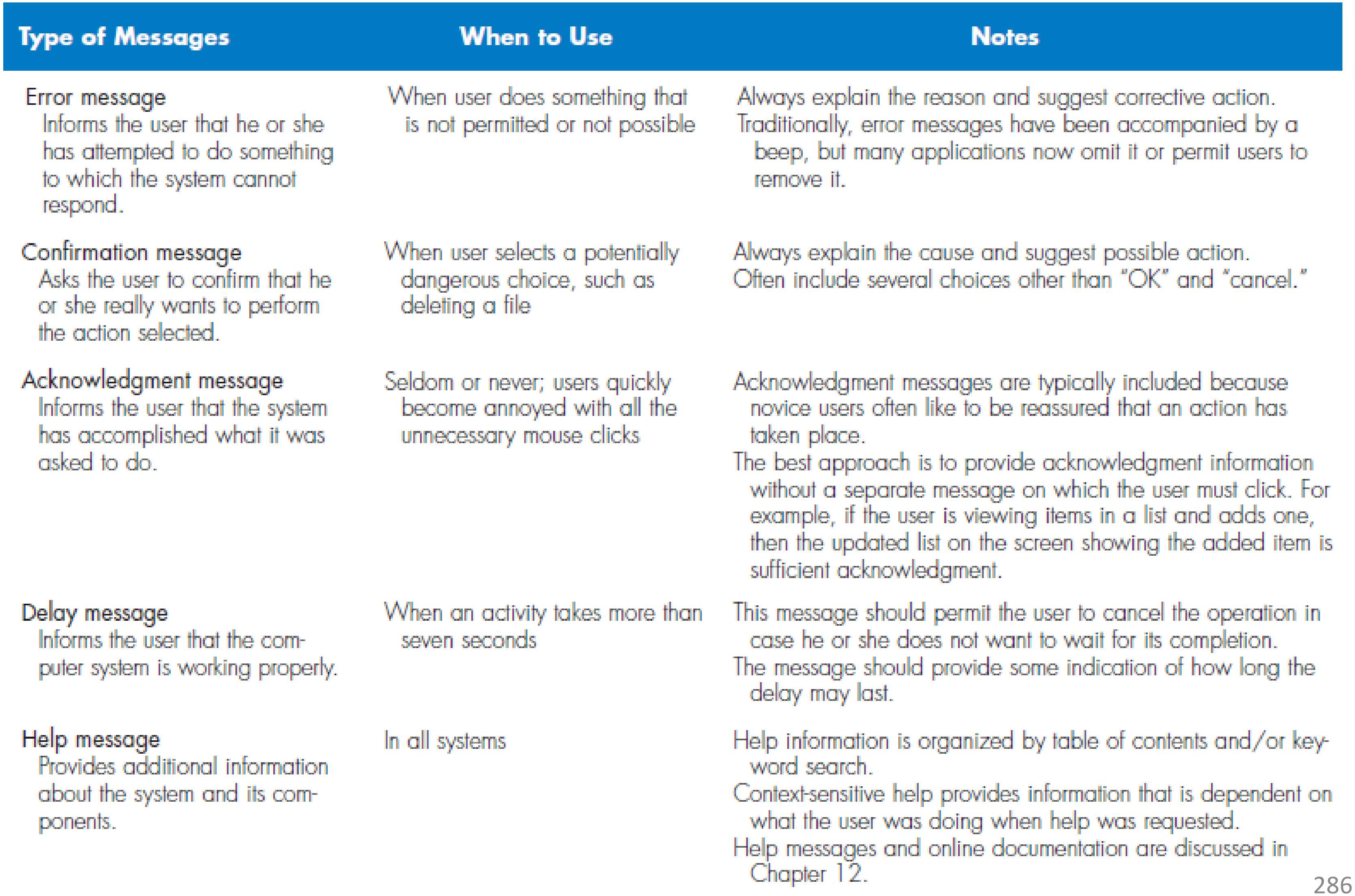
- Two traditional hardware for input:
  - Keyboard
  - Pointing device
- Three basic software approaches
  - Languages
  - Menus
  - Direct Manipulation

Navigation design

Messages

- The system responds to a user
- Informs about the status of the interaction.
- Can be:
  - Error
  - Confirmation
  - Acknowledgment
  - Delay
  - Help

Navigation design



Type of Messages	When to Use	Notes
Error message Informs the user that he or she has attempted to do something to which the system cannot respond.	When user does something that is not permitted or not possible	Always explain the reason and suggest corrective action. Traditionally, error messages have been accompanied by a beep, but many applications now omit it or permit users to remove it.
Confirmation message Asks the user to confirm that he or she really wants to perform the action selected.	When user selects a potentially dangerous choice, such as deleting a file	Always explain the cause and suggest possible action. Often include several choices other than "OK" and "cancel."
Acknowledgment message Informs the user that the system has accomplished what it was asked to do.	Seldom or never; users quickly become annoyed with all the unnecessary mouse clicks	Acknowledgment messages are typically included because novice users often like to be reassured that an action has taken place. The best approach is to provide acknowledgment information without a separate message on which the user must click. For example, if the user is viewing items in a list and adds one, then the updated list on the screen showing the added item is sufficient acknowledgment.
Delay message Informs the user that the computer system is working properly.	When an activity takes more than seven seconds	This message should permit the user to cancel the operation in case he or she does not want to wait for its completion. The message should provide some indication of how long the delay may last.
Help message Provides additional information about the system and its components.	In all systems	Help information is organized by table of contents and/or keyword search. Context-sensitive help provides information that is dependent on what the user was doing when help was requested. Help messages and online documentation are discussed in Chapter 12.

Input design

- Use Online and Batch Processing Appropriately
- Capture Data at the Source
- Minimize Keystrokes
- Types of Inputs:
  - Text
  - Numbers
  - Selection
- Input validation
  - Completeness, format, range, check digit, consistency and database check.

Output design

- Understand Report Usage
- Manage Information Load
- Minimize Bias
- Types of Outputs
- Media
    - Paper
    - Electronic

THANKS FOR ATTENTION!

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

EFOP-3.4.3-16-2016-00009

A felsőfokú oktatás minőségének és hozzáférhetőségének együttes javítása a Pannon Egyetemen

SYSTEM ANALYSIS AND SYSTEM DESIGN

9. PROGRAM DESIGN

Imre Budai

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

Objectives

- Be able to revise logical DFDs into physical DFDs.
- Be able to create a structure chart.
- Be able to write a program specification.
- Be able to write instructions using pseudocode.
- Become familiar with event-driven programming.

Introduction

- Program design techniques are very important
  - Pre-existing code needs to be understood, organized, and pieced together.
  - it is still common for the project team to have to
    - Write some (if not all) code
    - Produce original programs that support the application logic of the system.
  - Things must be done:
    - Implementation decisions will be made
    - Determine how the processes of the system will be organized.
    - Program specifications are developed.

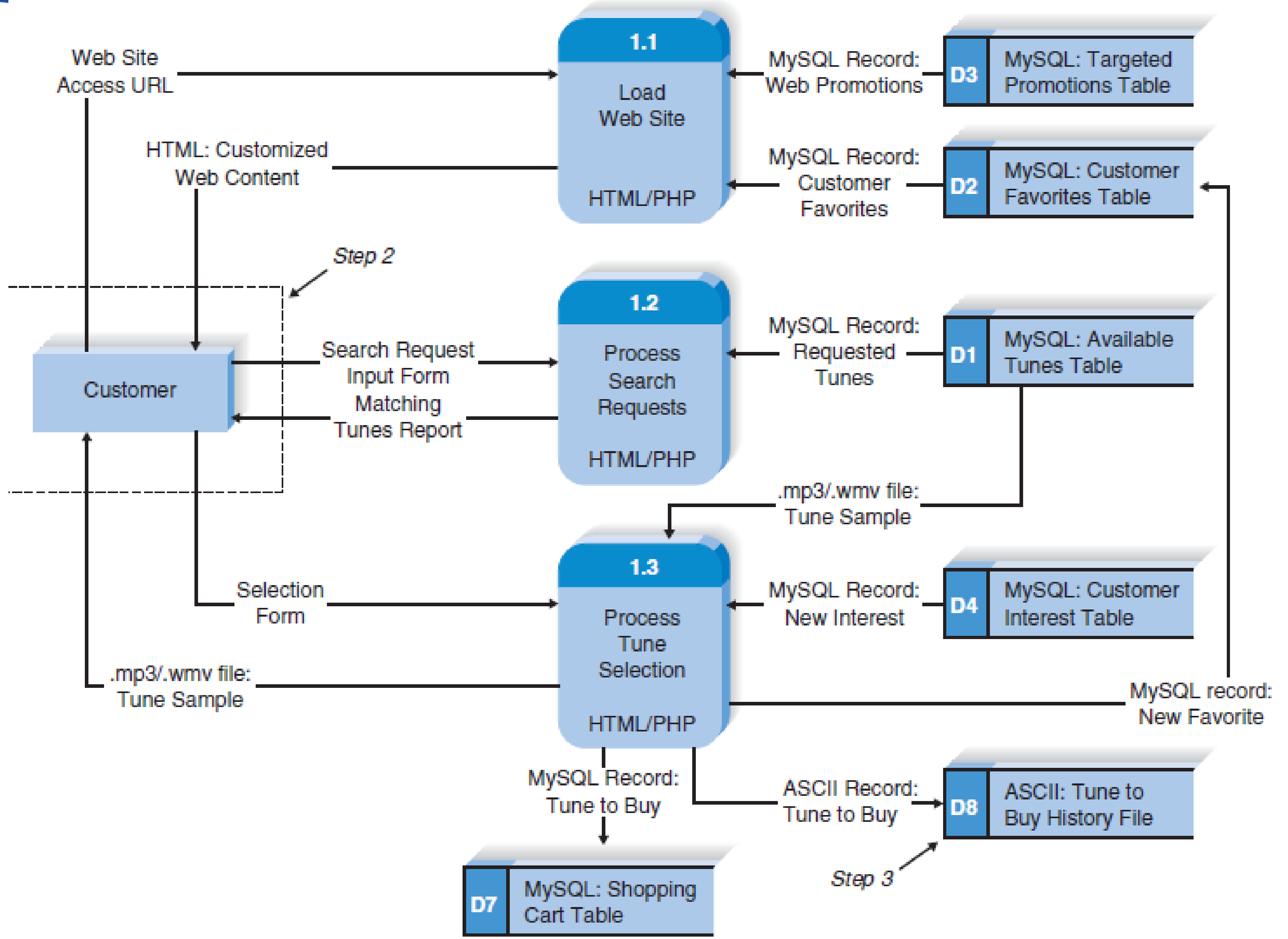
Moving from logical to physical

- Physical process models are created to show implementation details and explain how the final system will work.
    - References to actual technology
    - Format of information moving through processes
    - Human interactions
- These to-be models describe characteristics of the system that will be created, communicating the "systems view" of the new system.

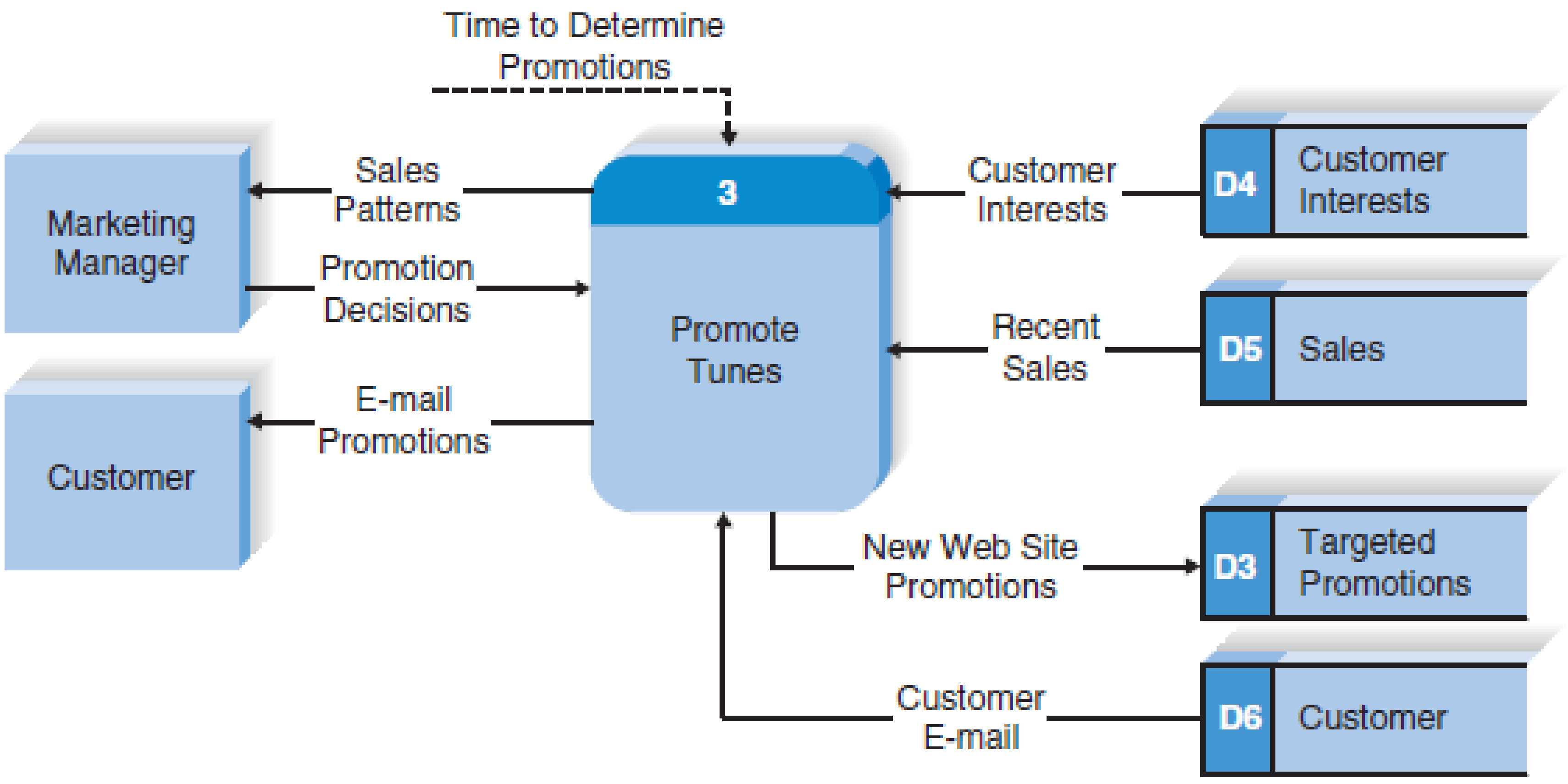
Physical Data Flow Diagram

- Contains the same components as the logical DFD.
- The same rules apply.
- But contains additional details.
- There are five steps to perform to make the transition to the physical DFD:
  1. Add Implementation References
  2. Draw a Human-Machine Boundary
  3. Add System-Related Data Stores, Data Flows, and Processes
  4. Update the Data Elements in the Data Flows
  5. Update the Metadata in the Computer-Aided Software Engineering Repository

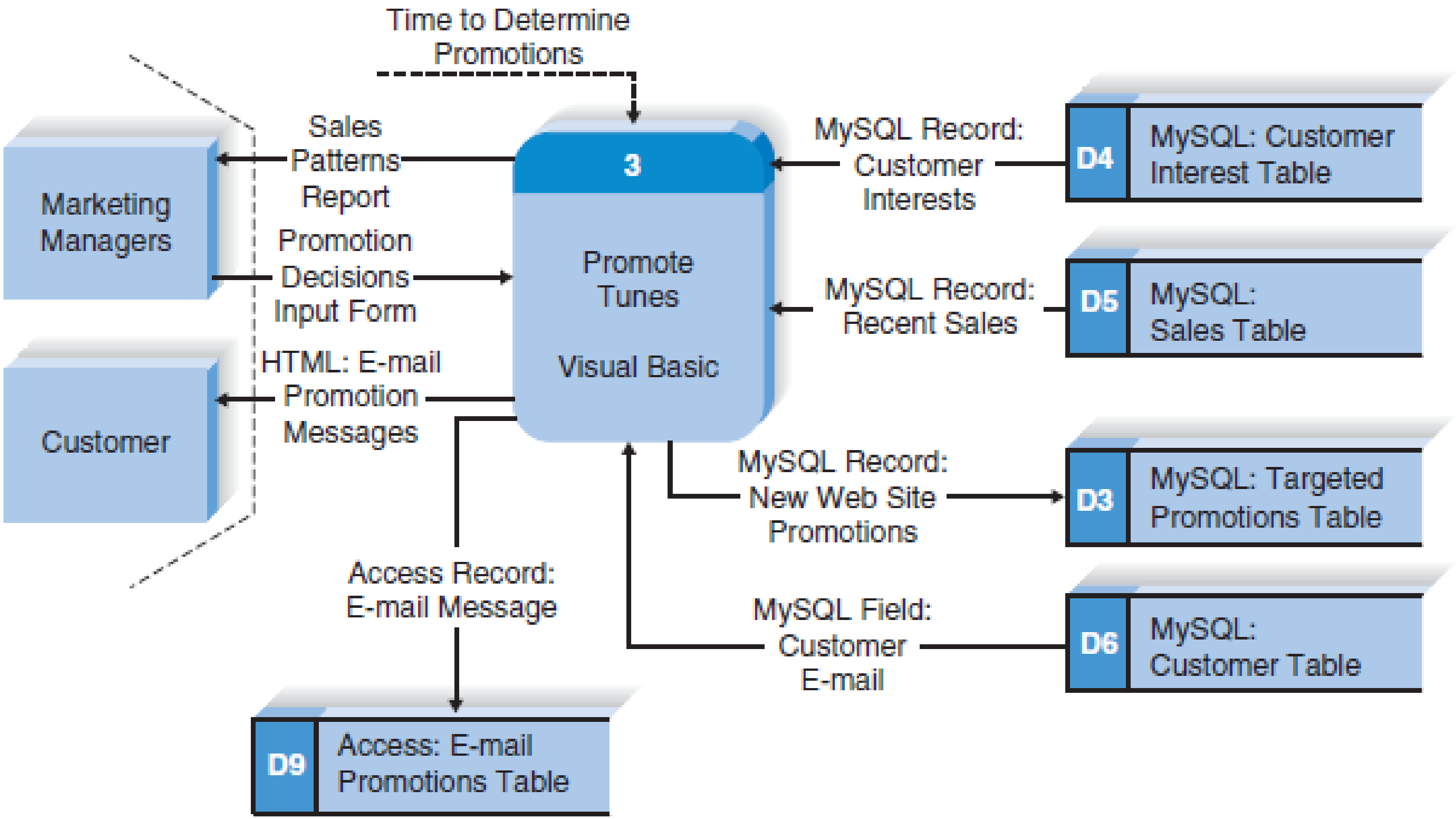
Physical Data Flow Diagram


1.1
Load Web Site
HTML/PHP
Web Site Access URL
HTML: Customized Web Content
MySQL Record: Web Promotions
D3
MySQL: Targeted Promotions Table
MySQL Record: Customer Favorites
D2
MySQL: Customer Favorites Table
Step 2
Customer
Search Request Input Form
Matching Tunes Report
1.2
Process Search Requests
HTML/PHP
MySQL Record: Requested Tunes
D1
MySQL: Available Tunes Table
.mp3/.wmv file: Tune Sample
1.3
Process Tune Selection
HTML/PHP
Selection Form
.mp3/.wmv file: Tune Sample
MySQL Record: New Interest
D4
MySQL: Customer Interest Table
MySQL record: New Favorite
MySQL Record: Tune to Buy
ASCII Record: Tune to Buy
D8
ASCII: Tune to Buy History File
Step 3
D7
MySQL: Shopping Cart Table

Physical Data Flow Diagram

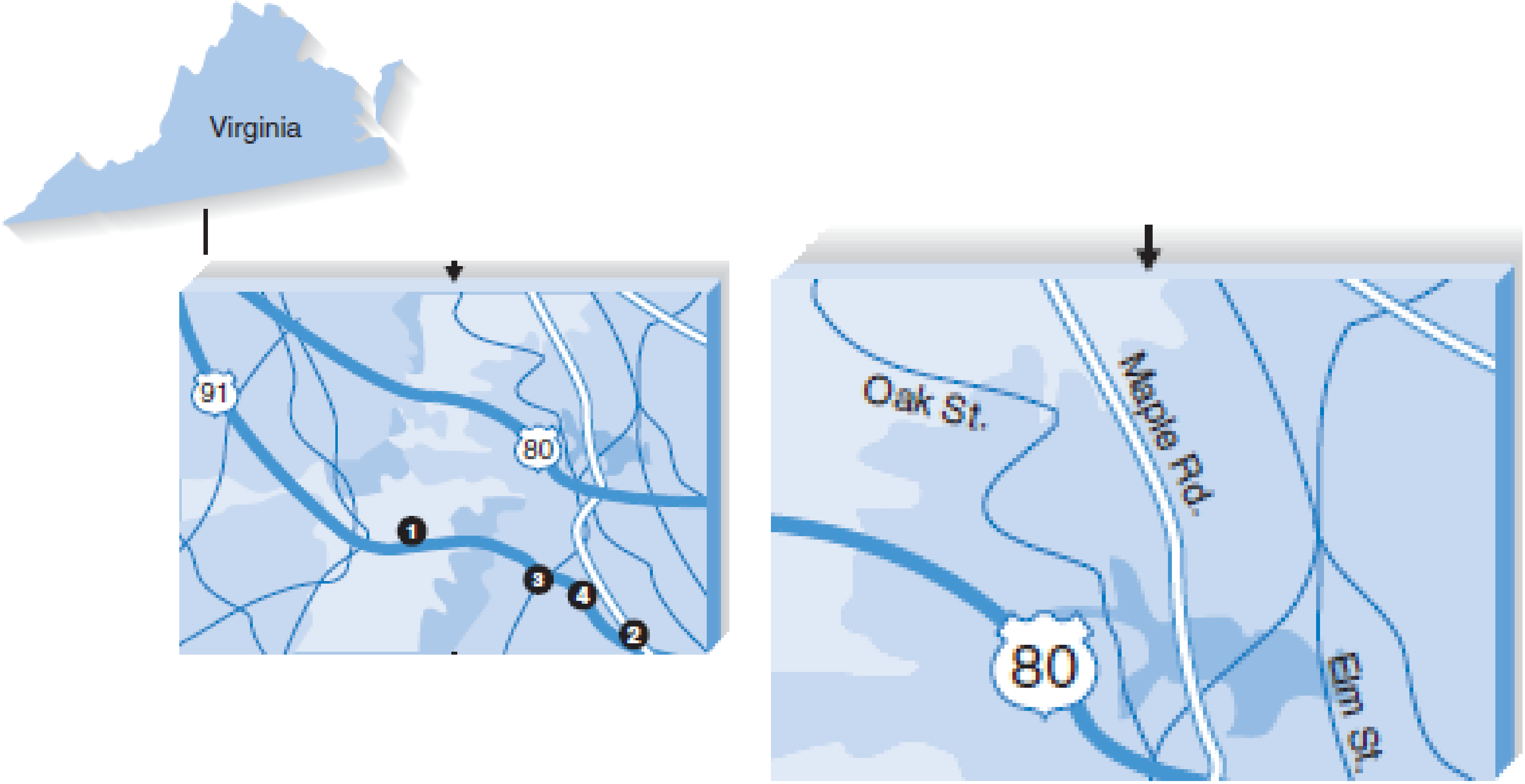


Physical Data Flow Diagram


Time to Determine Promotions
Sales Patterns Report
Marketing Managers
Promotion Decisions Input Form
Customer
HTML: E-mail Promotion Messages
3
Promote Tunes
Visual Basic
MySQL Record: Customer Interests
D4
MySQL: Customer Interest Table
MySQL Record: Recent Sales
D5
MySQL: Sales Table
MySQL Record: New Web Site Promotions
D3
MySQL: Targeted Promotions Table
Access Record: E-mail Message
MySQL Field: Customer E-mail
D6
MySQL: Customer Table
D9
Access: E-mail Promotions Table

Designing Programs

- Designing programs in a top-down modular approach



Designing Programs

- Good program design is similar to this top-down modular approach.
- Creating a high-level diagram that shows the various components of a

  program.
  - How the components should be organized.
  - How the components interrelate.
  - It is known as the structure chart.

    It illustrates the organization and interactions of the different pieces of code within the program to the analysts and programmers so that the program can be developed by many programmers working independently.

Structure chart

- It helps the analyst design the program for the new system.
- It shows all the components of code that must be included in a program at a high level.
- It is arranged in a hierarchical format that implies *sequence, selection* and *iteration*.
- It is read from top to bottom, left to right.
- It is numbered by a hierarchical numbering scheme.

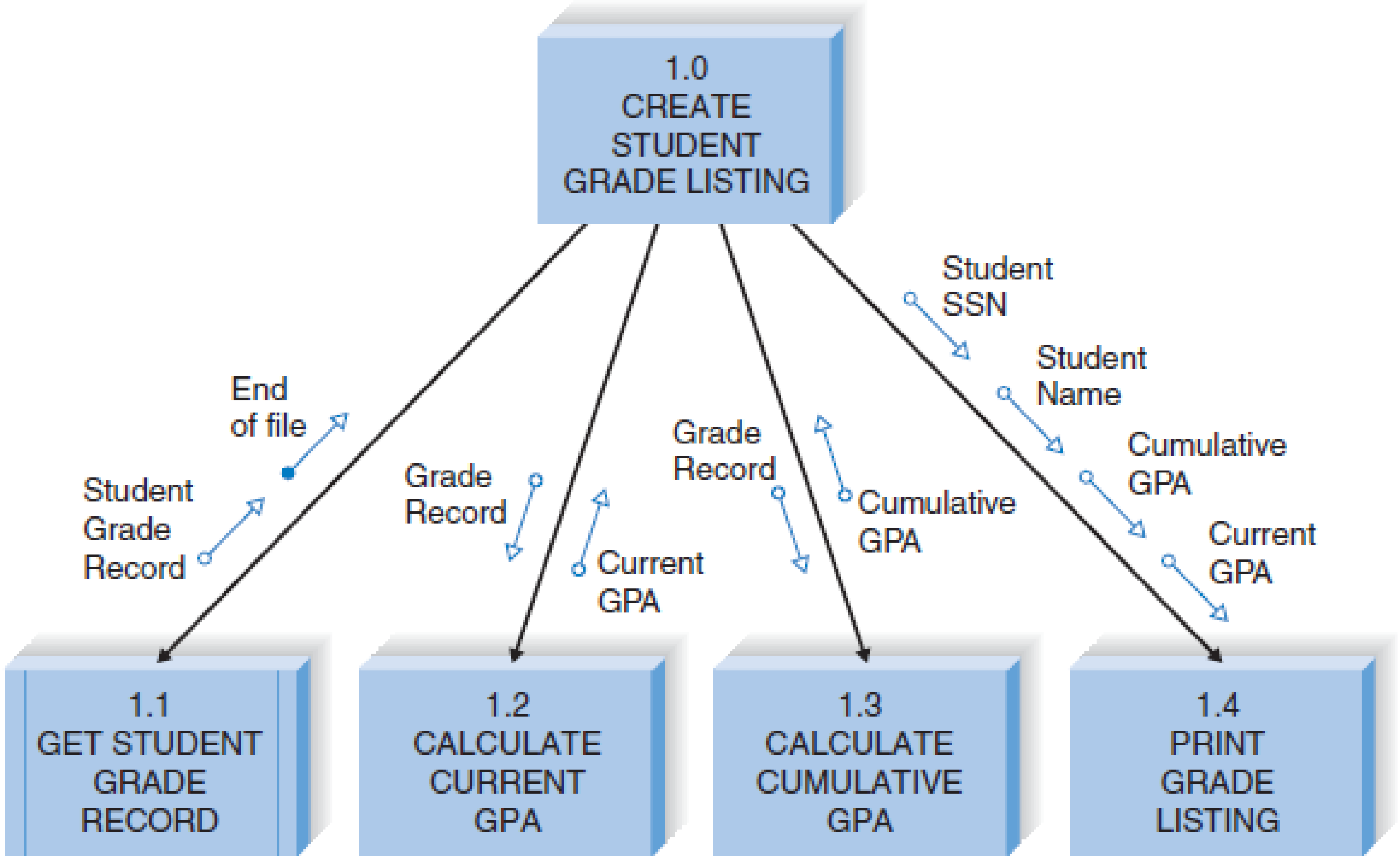
Structure chart

Syntax

- Modules
    - Lines of program code that perform a single function
    - Rectangles
- Lines, which represent the passing of control
- Couples
- Control module is a higher-level component that contains the logic for performing other modules

Structure chart

- Structure Chart Example (GPA – grade point average)



Structure chart

- Reused modules called library modules, have vertical lines on both sides – *they will appear several times on the structure chart.*
- Two special control
  - Loop – repeating
  - Conditional
- Data couples (empty)
- Control couples (filled)

Structure chart

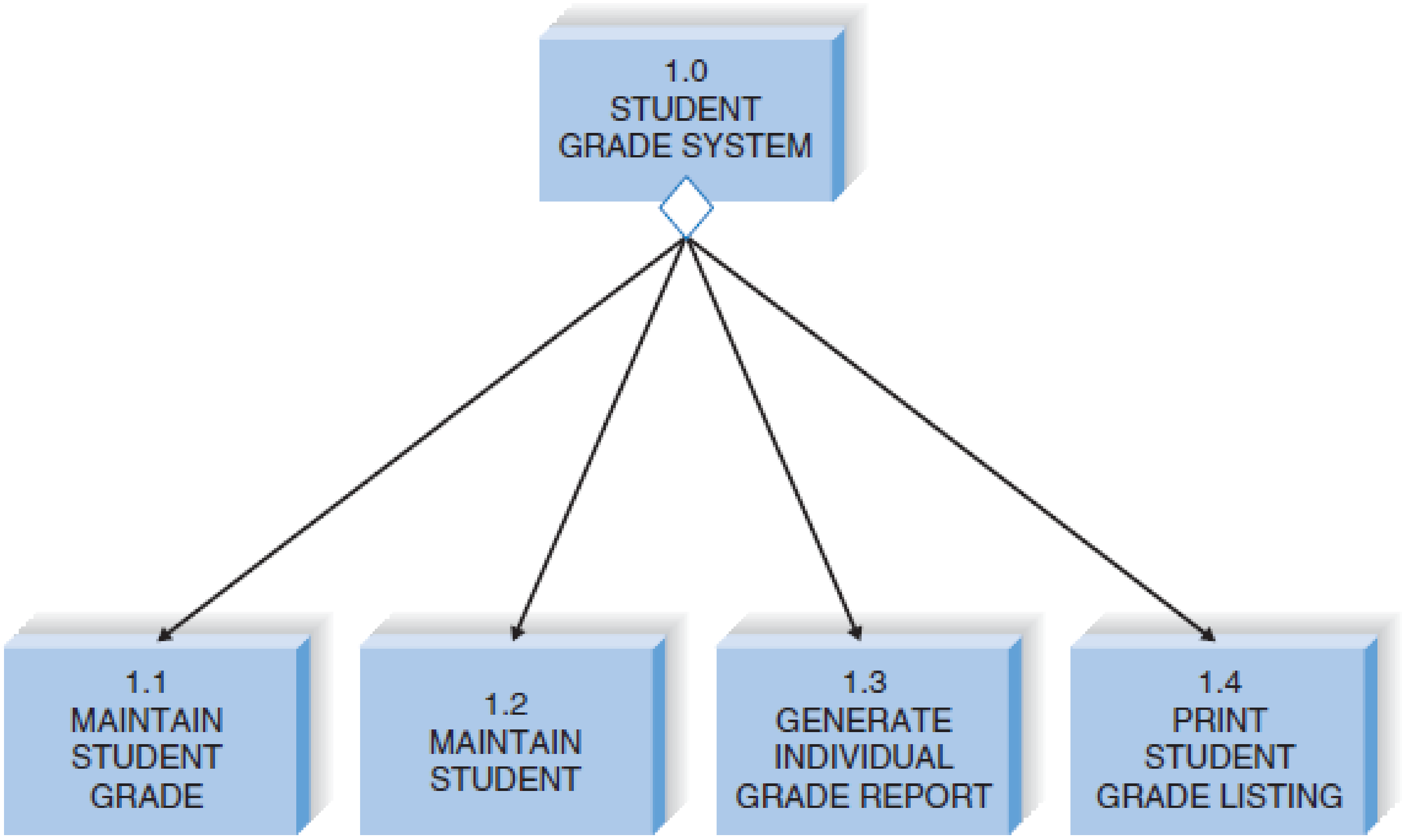
Building a structure chart

- Many times, process models are used as the starting point for structure charts.
    - Afferent processes: provide inputs into the system.
    - Central processes: perform critical functions in the operation.
    - Efferent processes: deal with system outputs.
- Each process of a DFD tends to represent one module on the structure chart.

Structure chart

- there are two basic arrangements, or structures, for combining structure chart modules:
  - Transaction Structure
    - Contains a control module that calls subordinate modules.
    - Each handles a particular transaction.
    - Usually the high levels of the DFD represent transaction structure.

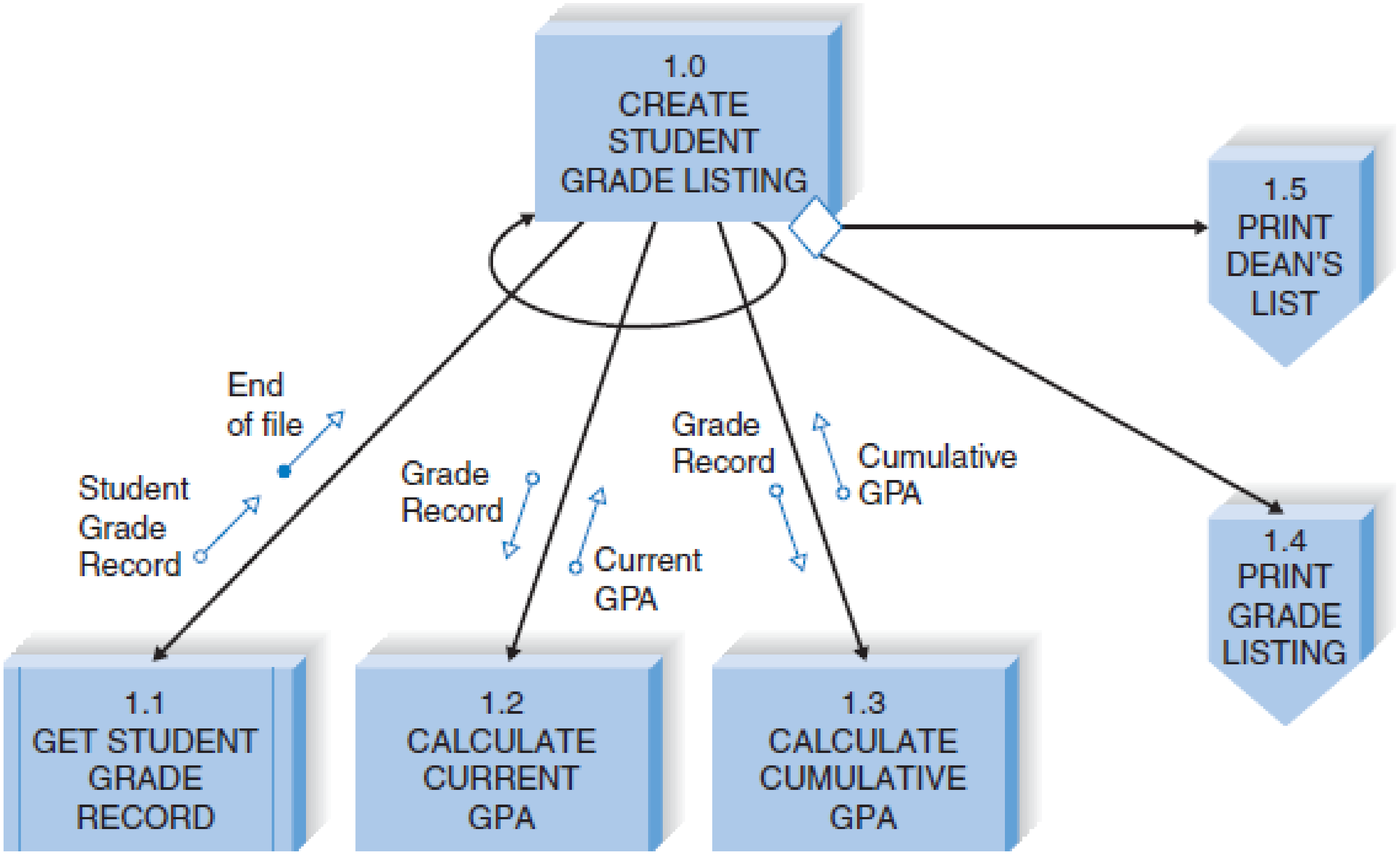
Structure chart


1.0 STUDENT GRADE SYSTEM
1.1 MAINTAIN STUDENT GRADE
1.2 MAINTAIN STUDENT
1.3 GENERATE INDIVIDUAL GRADE REPORT
1.4 PRINT STUDENT GRADE LISTING

Structure chart

- Transform Structure
  - Control module calls several subordinate modules in sequence
  - Often, each module accepts an input from the module preceding it.
  - Works on the input, then passes it to the next module for more processing.
  - The control module describes what the subordinates will do.
  - The subordinates are invoked from left to right.
  - Usually the lowest levels represent transform structures.

Structure chart



Structure chart

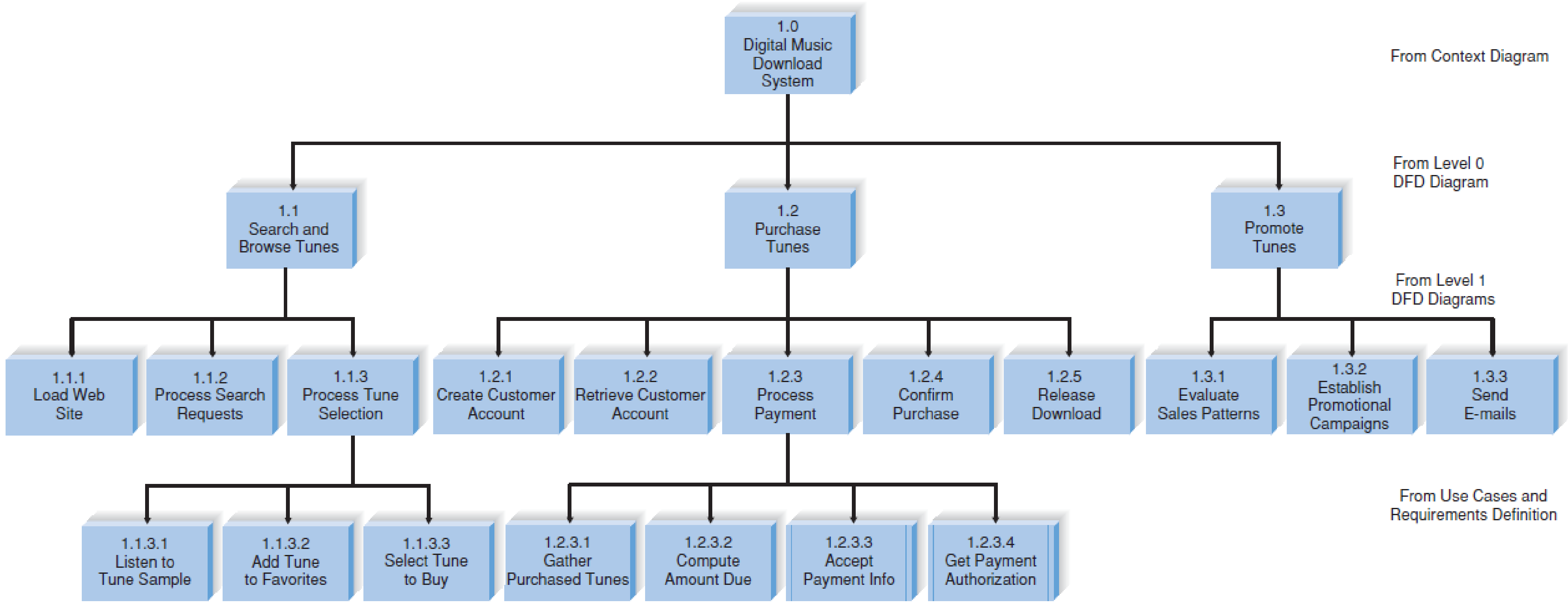
Example – Tune Source

- Step 1: Identify Modules and Levels
  - The context-level DFD (the overall system) is placed at the top of the structure chart.
  - The level 0 DFD processes are placed below it as subordinates.

    These particular structure of modules are transaction structures
    The subordinates represent different functions that can be called by the control module.
  - The level 1 DFD processes under search and browse tunes process control module are in transform structure.

    The subordinate modules are carried out in a sequence to perform the process that is represented by the control module, search and browse tunes.

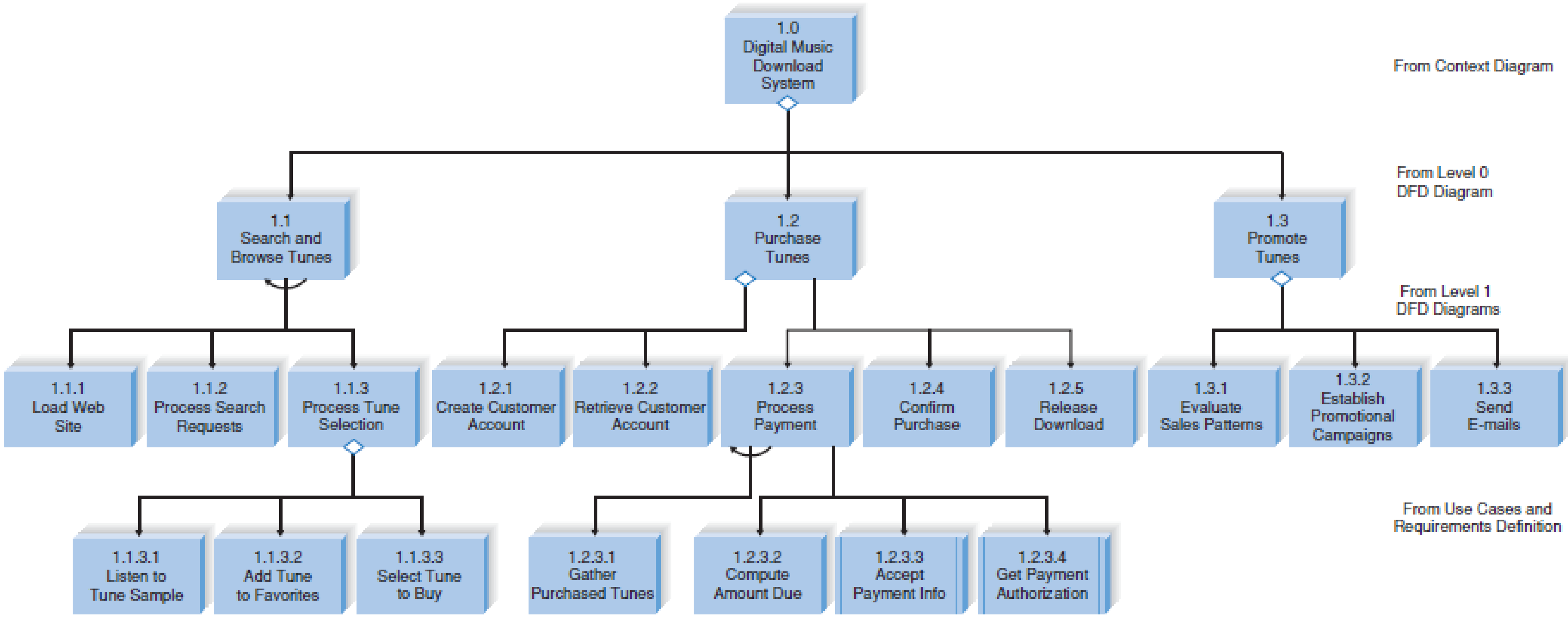
Structure chart


1.0 Digital Music Download System
From Context Diagram
1.1 Search and Browse Tunes
1.2 Purchase Tunes
1.3 Promote Tunes
From Level 0 DFD Diagram
From Level 1 DFD Diagrams
1.1.1 Load Web Site
1.1.2 Process Search Requests
1.1.3 Process Tune Selection
1.2.1 Create Customer Account
1.2.2 Retrieve Customer Account
1.2.3 Process Payment
1.2.4 Confirm Purchase
1.2.5 Release Download
1.3.1 Evaluate Sales Patterns
1.3.2 Establish Promotional Campaigns
1.3.3 Send E-mails
From Use Cases and Requirements Definition
1.1.3.1 Listen to Tune Sample
1.1.3.2 Add Tune to Favorites
1.1.3.3 Select Tune to Buy
1.2.3.1 Gather Purchased Tunes
1.2.3.2 Compute Amount Due
1.2.3.3 Accept Payment Info
1.2.3.4 Get Payment Authorization

Structure chart

- Step 2: Identify Special Connections
  - Add loops and conditional lines to represent modules that are repeated or optional.
  - A diamond is placed below a control module that directs subordinates, which may or may not be performed.
    - Customers may choose to listen to a tune sample, add it to the favourites list, or buy it.
    - They do not necessarily use all three alternatives.

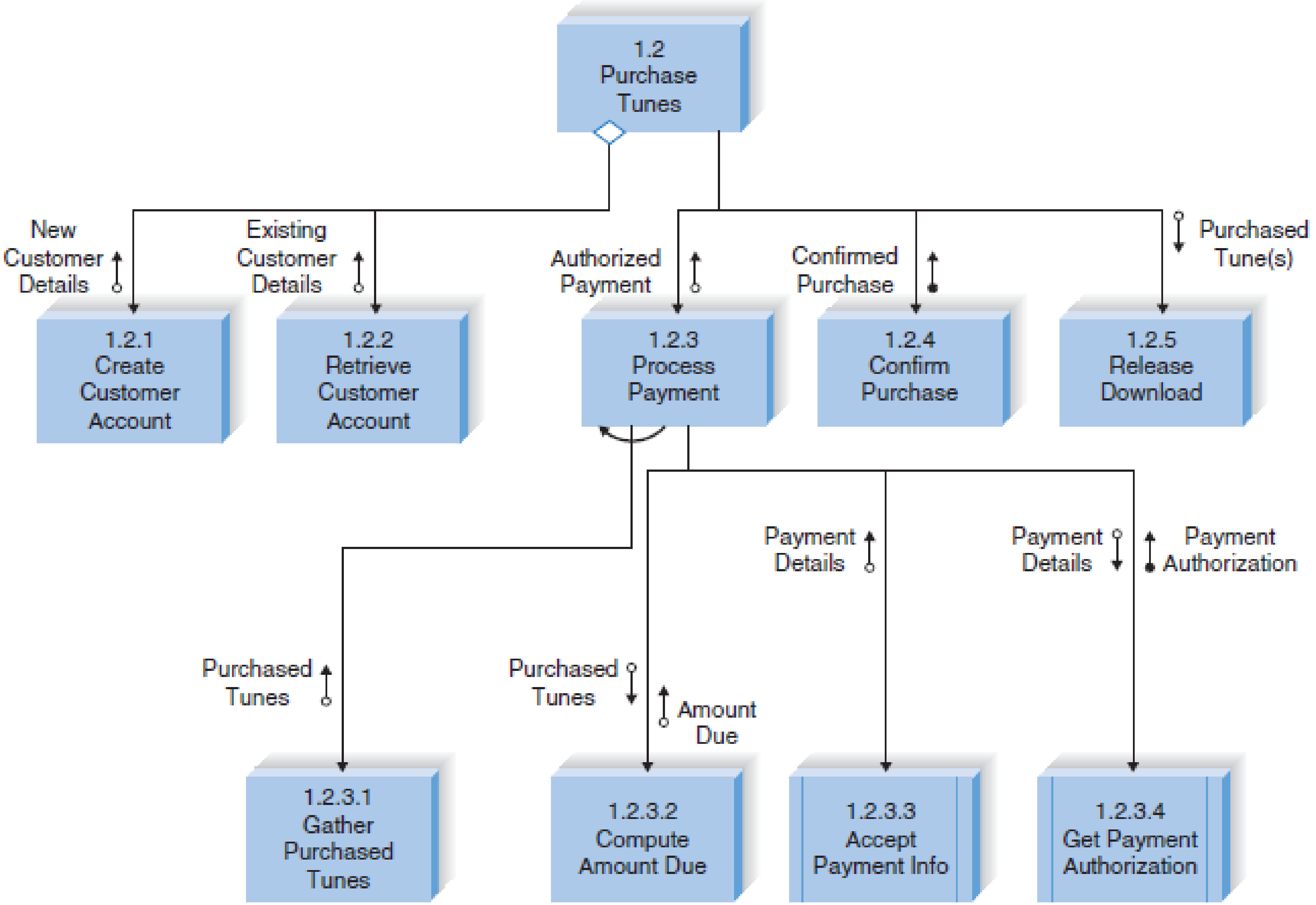
Structure chart


1.0 Digital Music Download System
From Context Diagram
1.1 Search and Browse Tunes
1.2 Purchase Tunes
1.3 Promote Tunes
From Level 0 DFD Diagram
From Level 1 DFD Diagrams
1.1.1 Load Web Site
1.1.2 Process Search Requests
1.1.3 Process Tune Selection
1.2.1 Create Customer Account
1.2.2 Retrieve Customer Account
1.2.3 Process Payment
1.2.4 Confirm Purchase
1.2.5 Release Download
1.3.1 Evaluate Sales Patterns
1.3.2 Establish Promotional Campaigns
1.3.3 Send E-mails
1.1.3.1 Listen to Tune Sample
1.1.3.2 Add Tune to Favorites
1.1.3.3 Select Tune to Buy
1.2.3.1 Gather Purchased Tunes
1.2.3.2 Compute Amount Due
1.2.3.3 Accept Payment Info
1.2.3.4 Get Payment Authoriz ation
From Use Cases and Requirements Definition

Structure chart

- Step 3: Add Couples
  - Information has to be identified that has to pass among the modules.
    - Data attributes
    - Control parameters
  - DFD provides us with some guidance.

Structure chart



Structure chart

- Step 4: Revise Structure Chart
  - Grey areas and decisions that need to be confirmed.
  - Tools that can help:
    - Process descriptions

      Are there any details of the processes that haven’t yet been captured on the diagram.
    - Data model

      Have the right records and specific fields been passed using the data couples.

Design Guidelines

- Build Modules with High Cohesion
  - Refers to how well the lines of code within each structure chart module relate to each other.
  - Ideally, a module should perform only one task.
    - Easy to understand.
    - Easy to build.
  - The more tasks that a module has to perform, the more complex the logic in the code must be to implement the tasks correctly.
  - Title of the module – using ‚and' means low cohesion.
  - Presence of control flags that passing down means low cohesion.
  - There are various types of cohesion.

Design Guidelines

- Cohesion types

1. **Functional** – module performs one problem related task.
2. **Sequential** – output from one task is used by the next.
3. **Communicational** – elements contribute to activities that use the same inputs or outputs.
4. **Procedural** – elements are performed in sequence but do not share data.
5. **Temporal** – activities are related in time.
6. **Logical** – list of activities; which one to perform is chosen outside of module.
7. **Coincidential** – no apparent relationship.

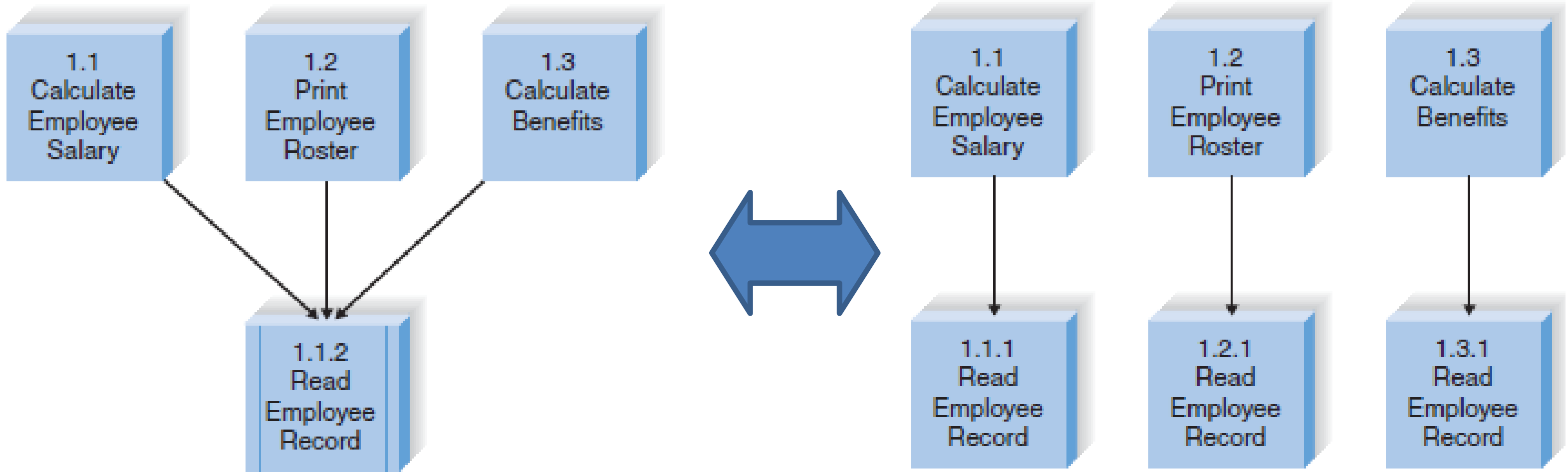
Design Guidelines

- Build Loosely Coupled Modules
  - Involves how closely modules are interrelated.
  - Modules are independent from each other.
  - Keeps code changes from rippling throughout the program.
  - Basically, the fewer the arrows on the diagrams, the easier it will be to make future alterations to the program.
  - Coupling types:
    1. **Data** – modules pass fields of data or messages.
    2. **Stamp** – modules pass record structures.
    3. **Control** – module passes control logic information.
    4. **Common** – modules refer to the same global data area.
    5. **Content** – module refers to the inside of another module.

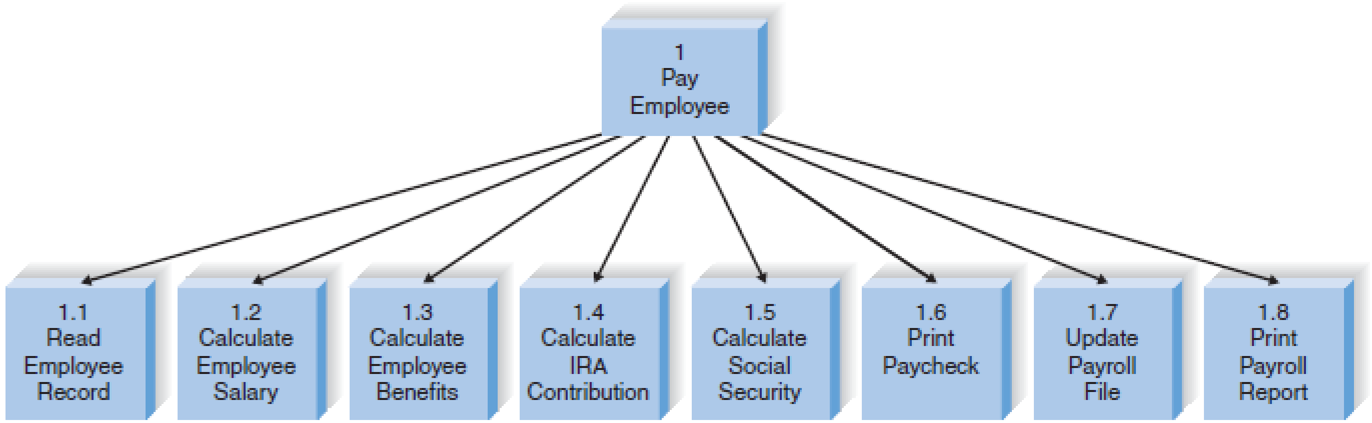
Design Guidelines

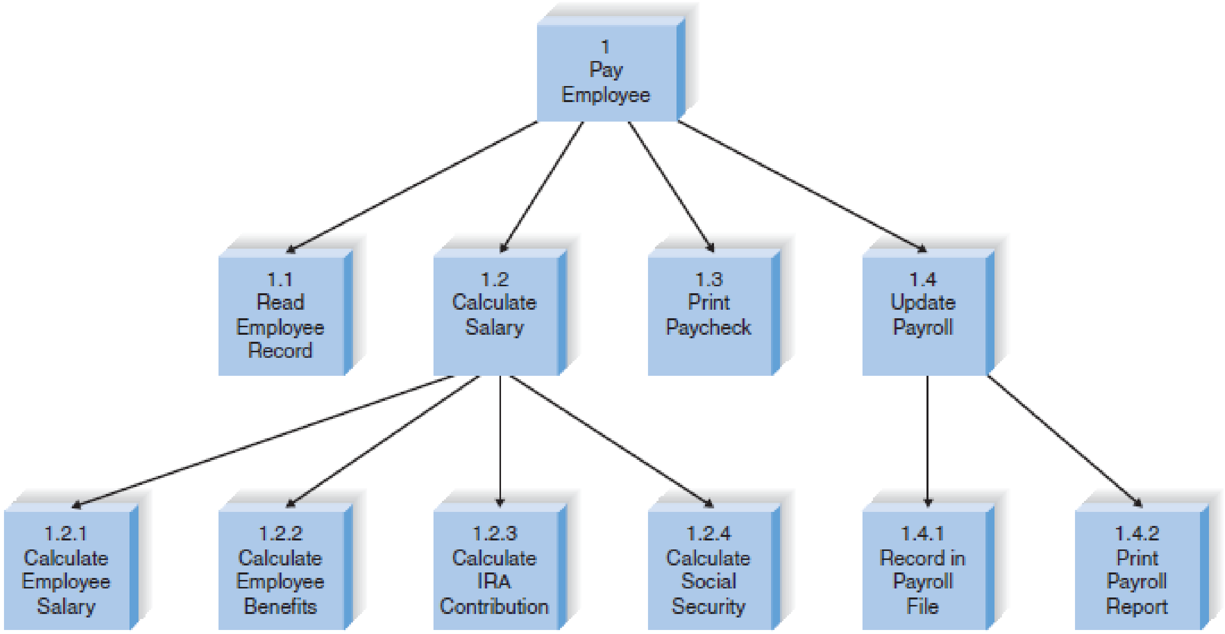
- Create High Fan-In
  - Describes the number of control modules that communicate with a subordinate
  - A module with high fan-in has many different control modules that call it.
  - Indicates that a module is reused in many places on the structure chart.
  - Suggests that the module contains well-written generic code.
- Avoid High Fan-Out
  - Avoid a large number of subordinates associated with a single control.

Design Guidelines


1.1 Calculate Employee Salary
1.2 Print Employee Roster
1.3 Calculate Benefits
1.1.2 Read Employee Record
1.1 Calculate Employee Salary
1.2 Print Employee Roster
1.3 Calculate Benefits
1.1.1 Read Employee Record
1.2.1 Read Employee Record
1.3.1 Read Employee Record

Design Guidelines


1 Pay Employee
1.1 Read Employee Record
1.2 Calculate Employee Salary
1.3 Calculate Employee Benefits
1.4 Calculate IRA Contribution
1.5 Calculate Social Security
1.6 Print Paycheck
1.7 Update Payroll File
1.8 Print Payroll Report


1 Pay Employee
1.1 Read Employee Record
1.2 Calculate Salary
1.3 Print Paycheck
1.4 Update Payroll
1.2.1 Calculate Employee Salary
1.2.2 Calculate Employee Benefits
1.2.3 Calculate IRA Contribution
1.2.4 Calculate Social Security
1.4.1 Record in Payroll File
1.4.2 Print Payroll Report

Design Guidelines

- Assess the Chart for Quality
  - Compile a checklist that may help you assess the quality of your structure chart.
    - Library modules have been created whenever possible.
    - The diagram has a high fan-in structure.
    - Control modules have no more than seven subordinates.
    - Each module performs only one function (high cohesion).
    - Modules sparingly share information (loose coupling).
    - Data couples that are passed are actually used by the accepting module.
    - Control couples are passed from "low to high."
    - Each module has a reasonable amount of code associated with it.

Program specification

- Program Information.
  - Name of the module.
  - Its purpose.
  - Deadline.
  - Target programming language.
- Events
  - Events that trigger the functionality in the program.
- Inputs and Outputs
- Pseudocode

THANKS FOR ATTENTION!

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

EFOP-3.4.3-16-2016-00009

A felsőfokú oktatás minőségének és hozzáférhetőségének együttes javítása a Pannon Egyetemen

SYSTEM ANALYSIS AND SYSTEM DESIGN

10. DATA STORAGE DESIGN

Imre Budai

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

Objectives

- Become familiar with several file and database formats.
- Describe several goals of data storage.
- Be able to revise a logical ERD into a physical ERD.
- Be able to optimize a relational database for data storage and data access.
- Become familiar with indexes.
- Be able to estimate the size of a database.

Introduction

- Four general functions:
  - Data storage
  - Data access logic
  - Application logic
  - Presentation logic
- The data storage function is concerned with how data is stored and handled by the programs that run the system.
- Avoid:
  - Inefficient systems
  - Long response times
  - Difficult-to-access information

Introduction

- There are two basic types of data storage formats
  - Files
  - Databases
- Physical data model
- Optimization for processing efficiency.
  - The final system is too slow.
  - Storage space.
  - Keep hardware costs down.
  - Availability, reliability, and security non-functional requirements.

Data Storage Formats

- Two main types of data storage formats:
  - Files

    Electronic lists of data that have been optimized to perform a particular transaction.
  - Databases

    Collection of groupings of information that are related to each other in some way.

    Database management system (DBMS) is software that creates and manipulates these databases.

Files

- Format of the files
  - Proprietary – older, legacy systems
  - Standard – newer systems
- Typically, files are organized sequentially.
- New records are added to the file’s end.
- Records can be associated with other records by a pointer.
- Sometimes files are called linked lists.

Files

- Types of files differ in the way they are used to support an application:
  - Master files
  - look-up files
  - Transaction files
  - Audit files
  - History files

Files

- Master file stores information that is:
  - Important to the business.
  - Kept for long periods.
  - New records are appended to the end of the file.
  - If changes need to be made to existing records, programs must be written to update the old information.
- Look-up files contain static values:
  - List of valid codes.
  - Names of the cities/counties/etc.
  - Typically used for validation.

Files

- Transaction file holds information for updating a master file.
  - Can be destroyed after successful update.
- Audit file records “before” and “after” images of data.
  - Audit can be performed if the integrity of the data is questioned.
- History file stores past transactions
  - They are no longer needed by system users.
  - Typically stored off-line.
  - Can be accessed on an as-needed basis.
  - Other files can then be streamlined to include only active or very recent information.

Databases

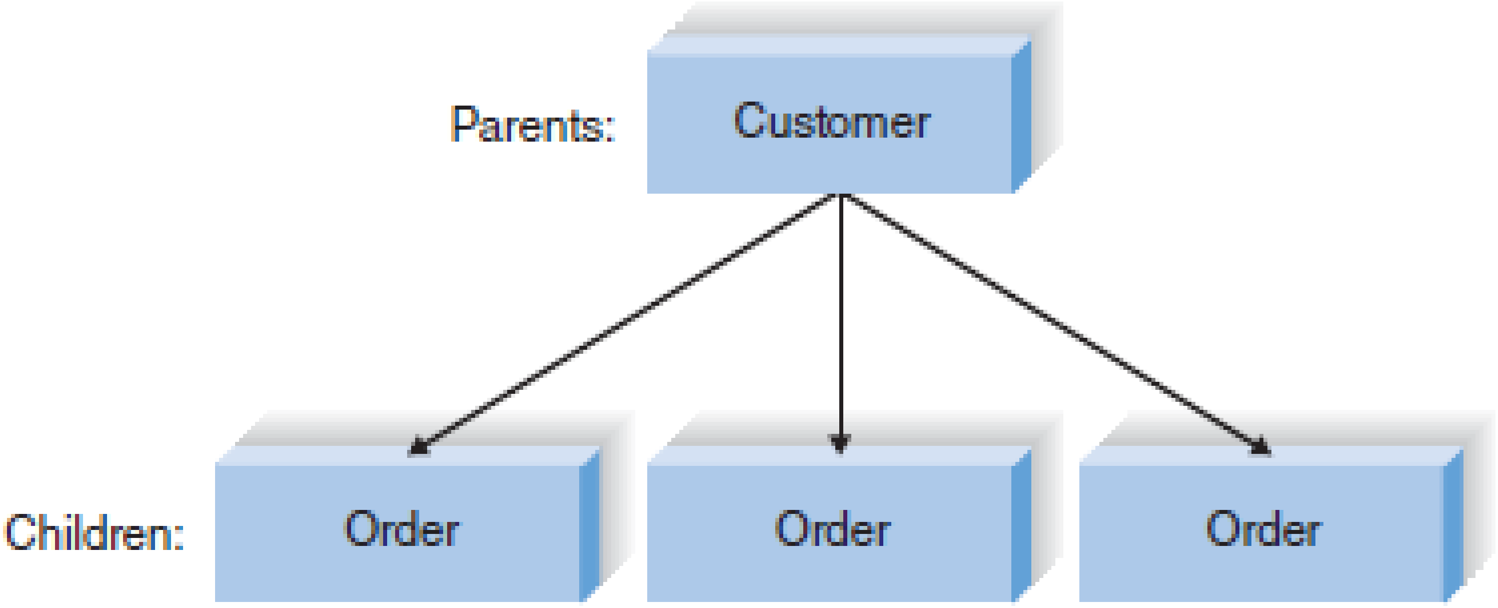
- There are many different types of on the market today.
  - Legacy
  - Relational
  - Object
  - Multidimensional
  - Non-relational
- We classify databases in terms of how they store and manipulate data.

Databases

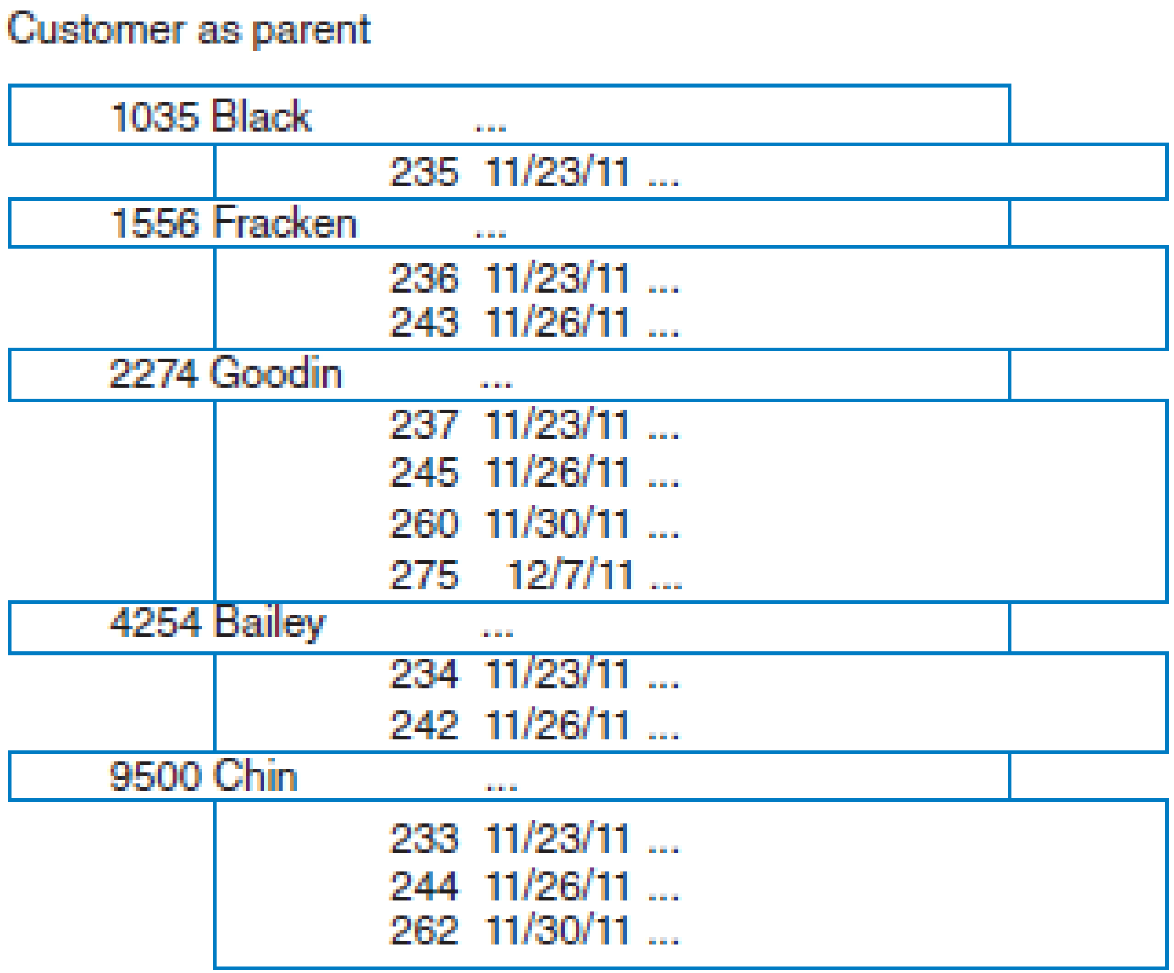
- Legacy Databases
  - Based on older, sometimes outdated technology that is seldom used to develop.
    - Maintenance.
    - Migration from existing system.
  - Hierarchical databases (like IDMS)
    - Inverted trees.
    - Represents relationships.
    - The record at the top of the tree has zero or more child records.
    - As parents for other records.

Databases

- Known for rapid search capabilities.
    - Used in the early systems in the airline industry



Databases



Customer as parent

1035 Black ...
235 11/23/11 ...
1556 Fracken ...
236 11/23/11 ...
243 11/26/11 ...
2274 Goodin ...
237 11/23/11 ...
245 11/26/11 ...
260 11/30/11 ...
275 12/7/11 ...
4254 Bailey ...
234 11/23/11 ...
242 11/26/11 ...
9500 Chin ...
233 11/23/11 ...
244 11/26/11 ...
262 11/30/11 ...

Databases

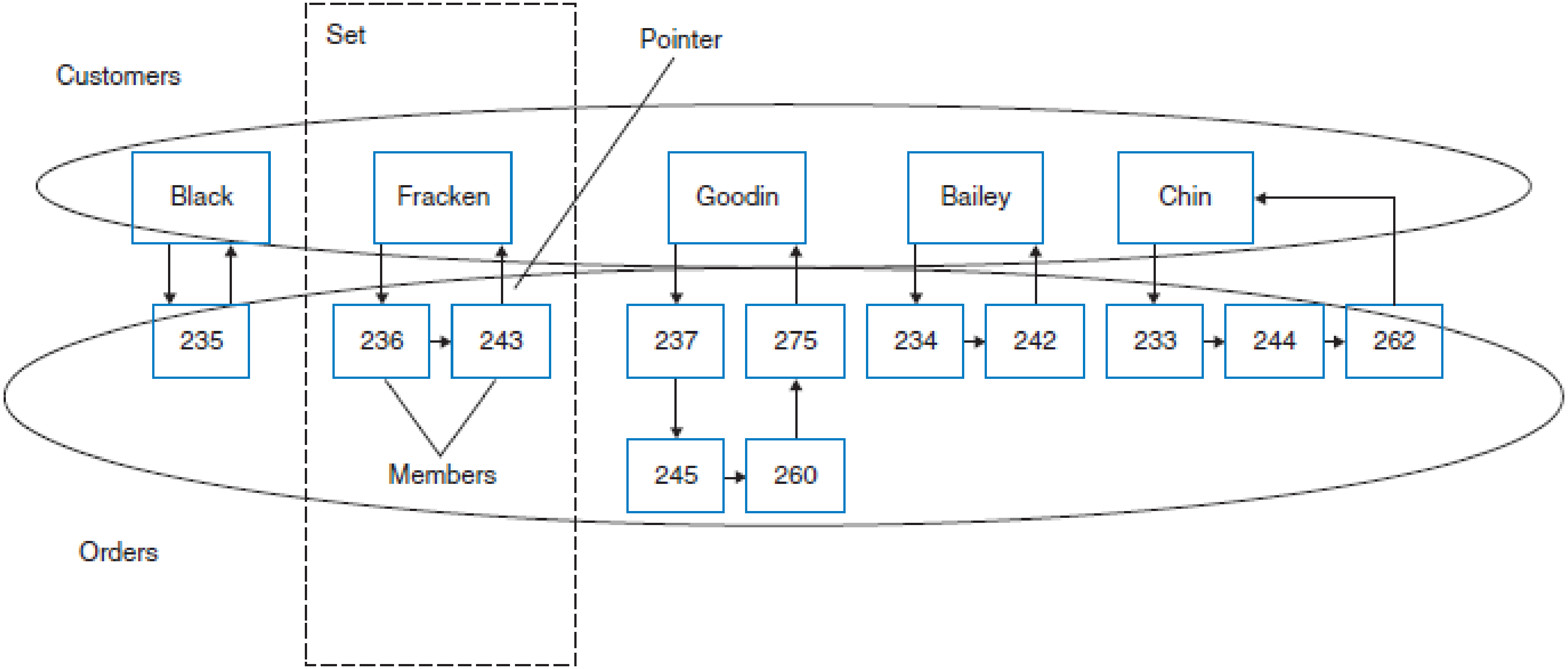
  - Cannot efficiently represent many-to-many relationships or non-hierarchical associations – a major drawback.

- Network databases
  - Collections of records.
  - Records are related to each other through pointers.
  - A record is a member in one or more sets.
  - The pointers link the members in a set together.

Databases

EFOP-3.4.3-16-2016-00009

A felsőfokú oktatás minőségének és hozzáférhetőségének együttes javítása a Pannon Egyetemen



Databases

- Both kinds of legacy systems can handle data quite efficiently.
- Require a great deal of programming effort.
- The application system software needs to contain code that manipulates the database pointers.
  - No separation between application and data access logic.
    - Structure of database.
    - Changes in structure.
    - Maintain the pointers.
    - Etc.

Databases

- Relational Databases
  - The most popular kind of databases.
  - Less “machine efficient”.
  - Much easier to work with from a development perspective.

  - Collection of tables.
    - Have a set number of columns.
    - Have a variable number of rows.
  - Primary (unique) key .
  - Foreign keys.
  - Supports referential integrity.

- Structured Query Language
  - The standard language for accessing the data in the tables.
  - Operates on complete tables.
  - When queries must include information from more than one table:
    - The tables first are joined together,
    - On the basis of their primary key and foreign key relationships,
    - And treated as if they were one large table.

Databases

- Object (oriented) Databases
  - Data treated as objects.
    - Data (attributes)
    - Processes (behaviours)
  - An object changes or accesses its own attributes only through its behaviours.
  - Objects may communicate with each other.
  - Changes to one object have no effect on other objects.
  - The combination of data and processes is represented by object classes – the major categories of objects in the system.
    - Subclasses / special cases.

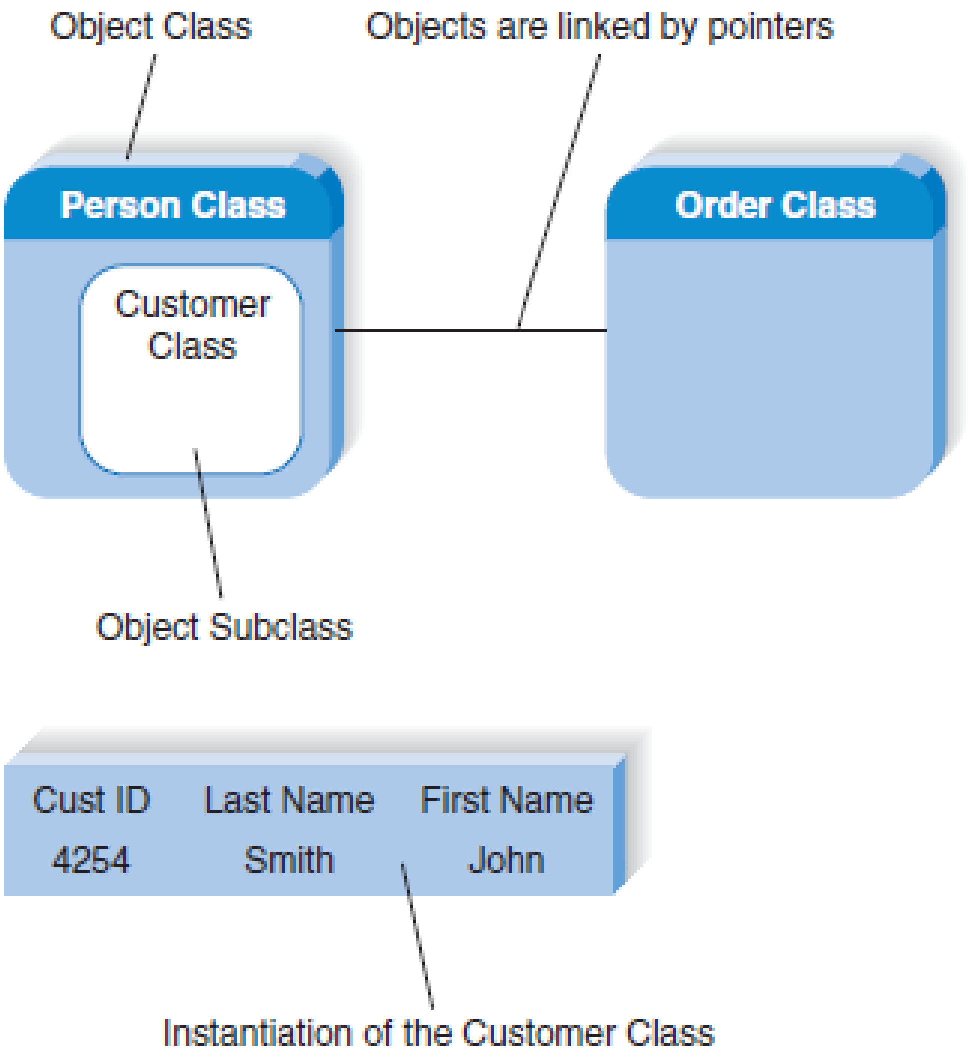
    Person class: subclasses of employee and customer

Databases

- Object-oriented database management systems mainly used for
  - Multimedia applications.
  - Systems that involve complex data.

    Telecommunications, financial services, health care, transportation
  - Popular technology for supporting:
  - E-commerce,
  - Online catalogs,
  - Large web multimedia applications.
- Hybrid OODBMS
  - Databases with both object and relational features.

Databases


Object Class
Objects are linked by pointers
Person Class
Customer Class
Order Class
Object Subclass
Instantiation of the Customer Class

Cust ID	Last Name	First Name
4254	Smith	John

Databases

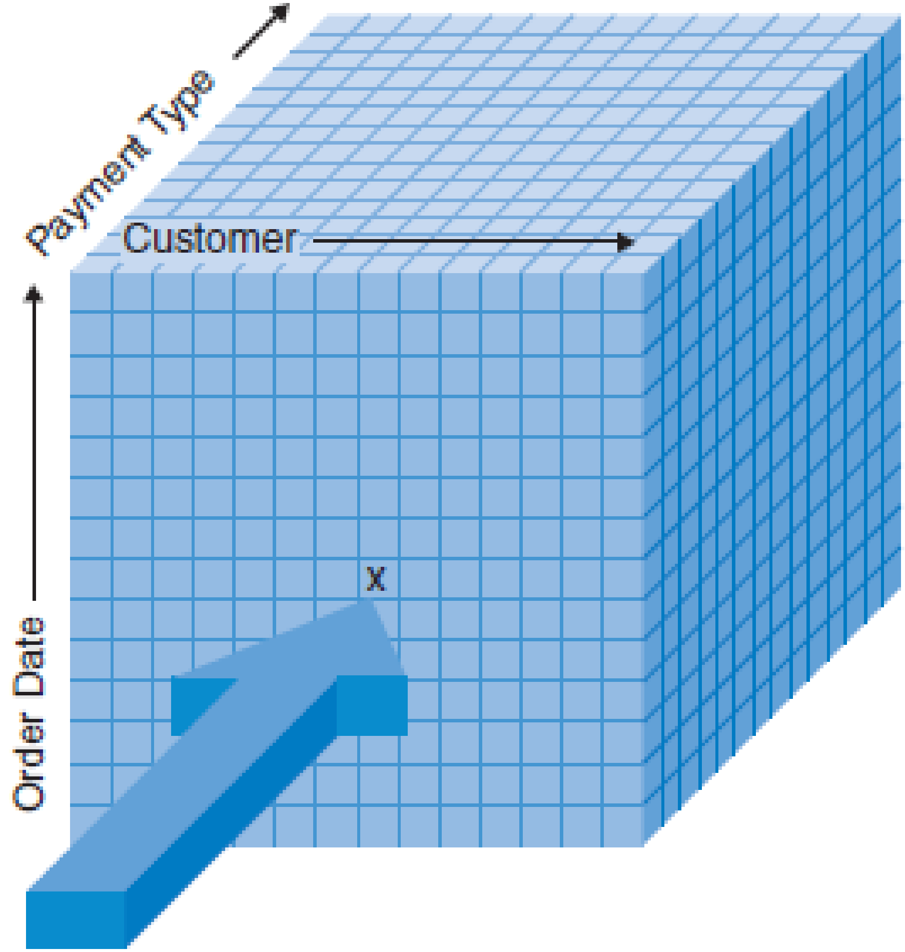
- Multidimensional Databases
  - Used in data warehousing.
    - Taking data from a company’s transaction processing systems.
    - Transforming the data (cleaning up, totaling, aggregating).
    - Storing the data for use in a data warehouse.
    - Supporting decision support systems.
    - Relies on relational technology as its storage format.
    - Data marts created for specific purposes.
    - Data mart receives downloads of data from the data warehouse regularly.
    - Data marts are usually created with multidimensional databases.

Databases

- Multidimensional Databases
  - Used in data warehousing.
    - Search for not a particular record.
    - But display information that is aggregated.
    - Legacy, object, and relational databases are designed and optimized to provide access to individual records – not to store data to support aggregations of data on multiple dimensions.

Databases

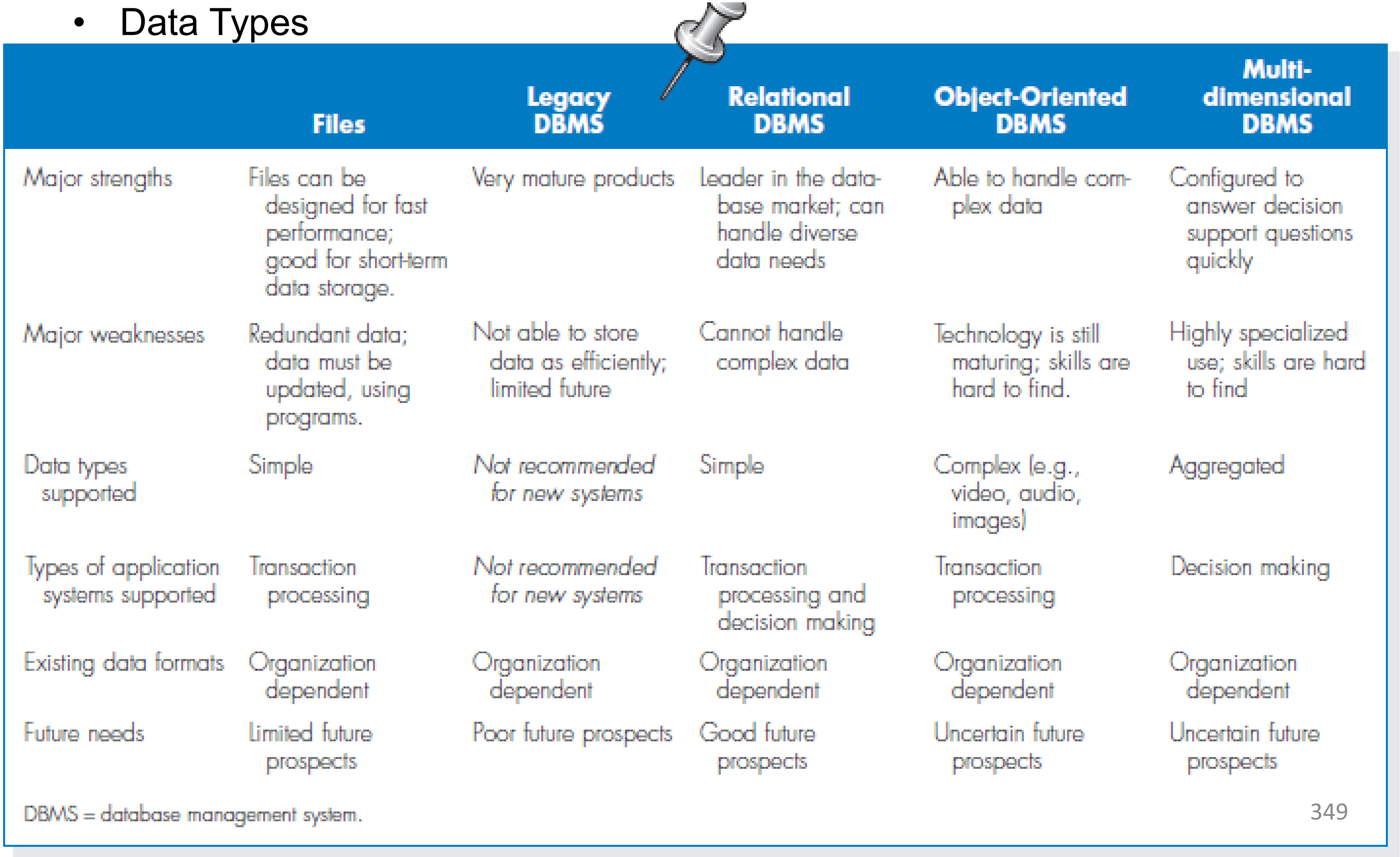
- Initial loading is slow.
- Data access is fast.



Last quarter, how many customers placed more than one order, using an American Express card?

Selecting a Storage Format

- Data Types



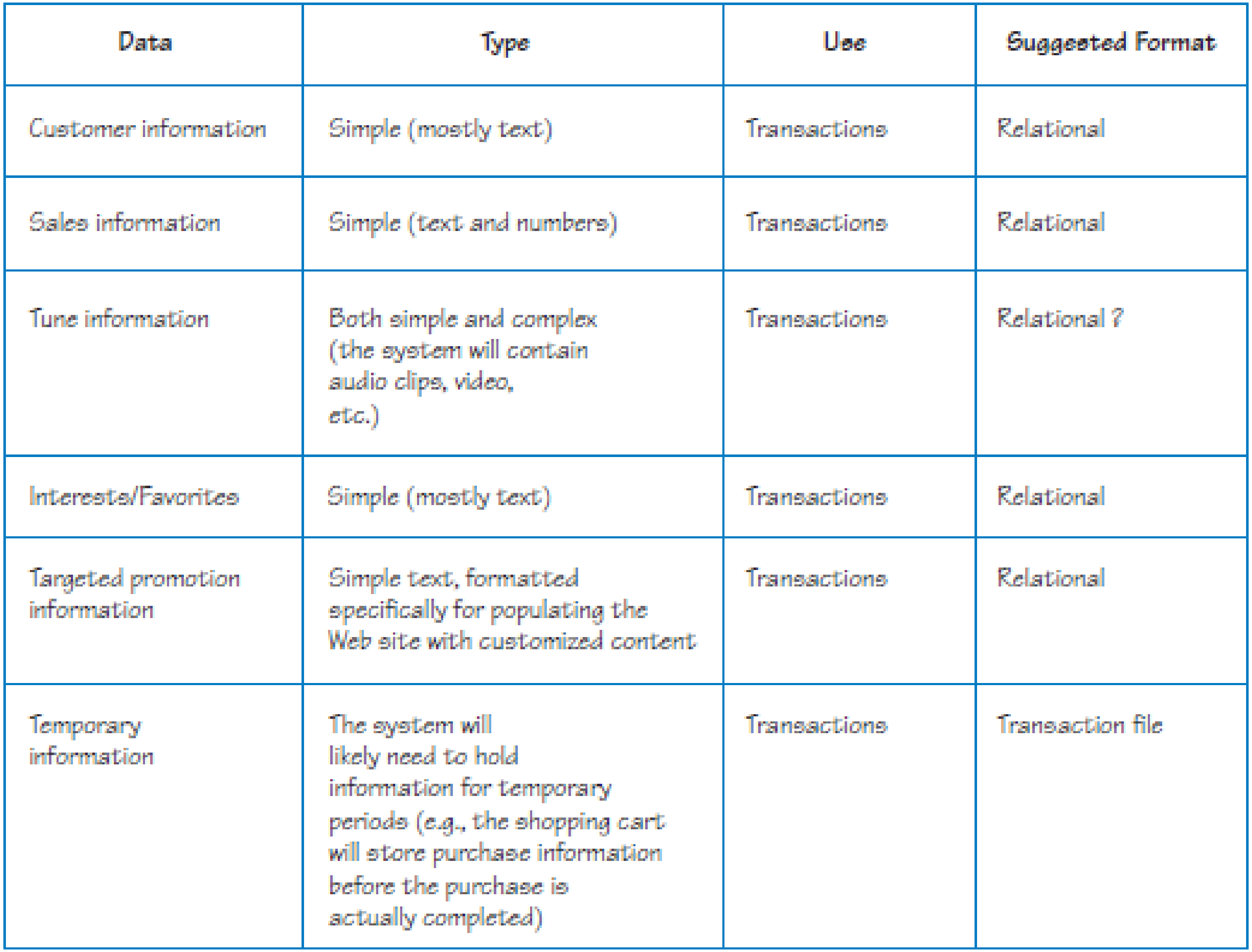
	Files	Legacy DBMS	Relational DBMS	Object-Oriented DBMS	Multi-dimensional DBMS
Major strengths	Files can be designed for fast performance; good for short-term data storage.	Very mature products	Leader in the database market; can handle diverse data needs	Able to handle complex data	Configured to answer decision support questions quickly
Major weaknesses	Redundant data; data must be updated, using programs.	Not able to store data as efficiently; limited future	Cannot handle complex data	Technology is still maturing; skills are hard to find.	Highly specialized use; skills are hard to find
Data types supported	Simple	*Not recommended for new systems*	Simple	Complex (e.g., video, audio, images)	Aggregated
Types of application systems supported	Transaction processing	*Not recommended for new systems*	Transaction processing and decision making	Transaction processing	Decision making
Existing data formats	Organization dependent	Organization dependent	Organization dependent	Organization dependent	Organization dependent
Future needs	Limited future prospects	Poor future prospects	Good future prospects	Uncertain future prospects	Uncertain future prospects

DBMS = database management system.

Selecting a Storage Format

- Type of Application System
  - Transaction processing systems.
  - Decision support systems.
- Existing Storage Formats
  - Consider the existing data storage formats in the organization.
  - Resources needed for training, for e.g.
- Future Needs

Tune Source



Data	Type	Use	Suggested Format
Customer information	Simple (mostly text)	Transactions	Relational
Sales information	Simple (text and numbers)	Transactions	Relational
Tune information	Both simple and complex (the system will contain audio clips, video, etc.)	Transactions	Relational ?
Interests/Favorites	Simple (mostly text)	Transactions	Relational
Targeted promotion information	Simple text, formatted specifically for populating the Web site with customized content	Transactions	Relational
Temporary information	The system will likely need to hold information for temporary periods (e.g., the shopping cart will store purchase information before the purchase is actually completed)	Transactions	Transaction file

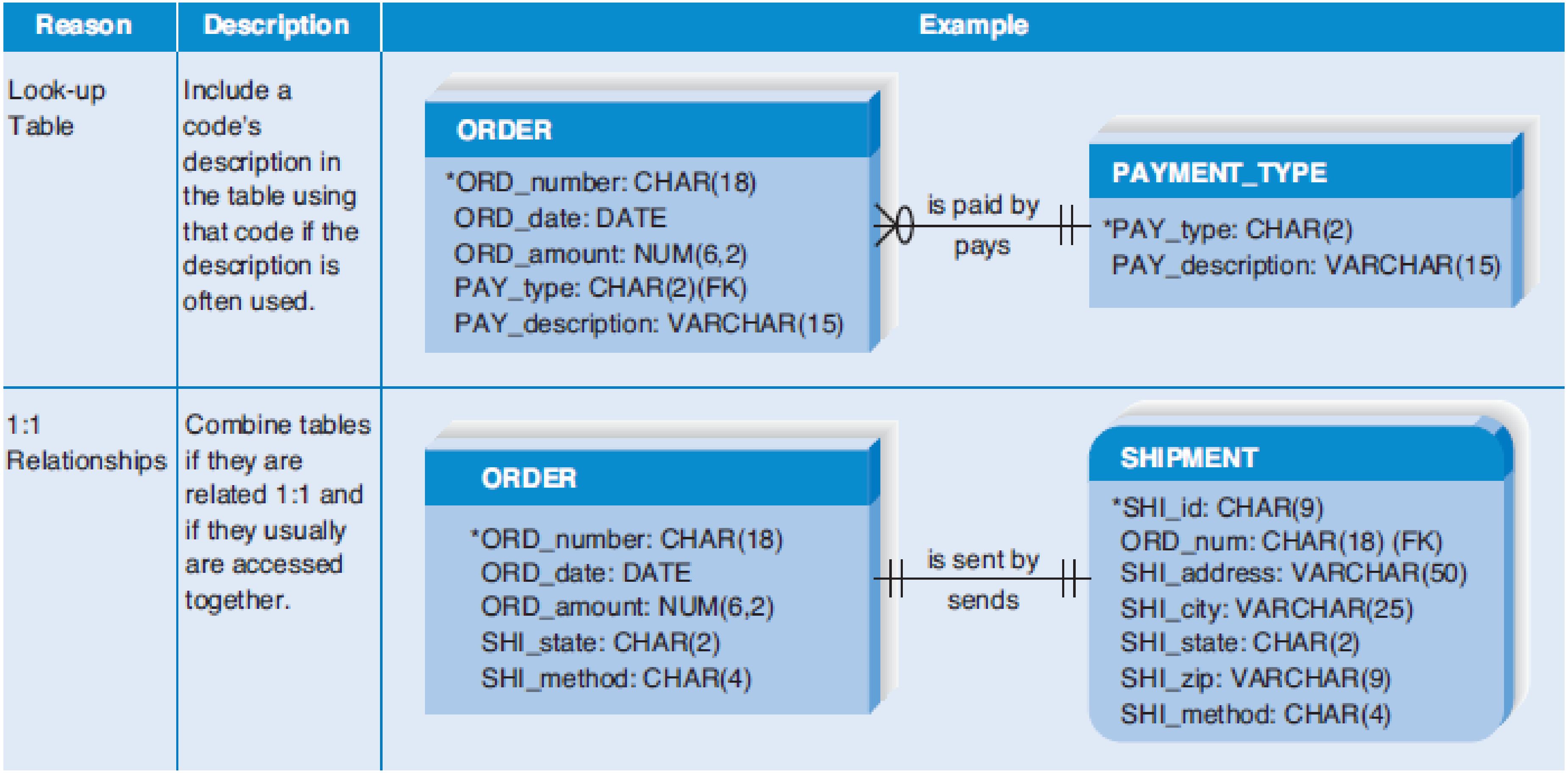
Moving to physical

- Physical Entity Relationship Diagram
  - Change Entities to Tables or Files
  - Change Attributes to Fields
  - Add Primary Keys
  - Add Foreign Keys
  - Add System-Related Components
  - Revisiting the CRUD Matrix

Optimizing data storage

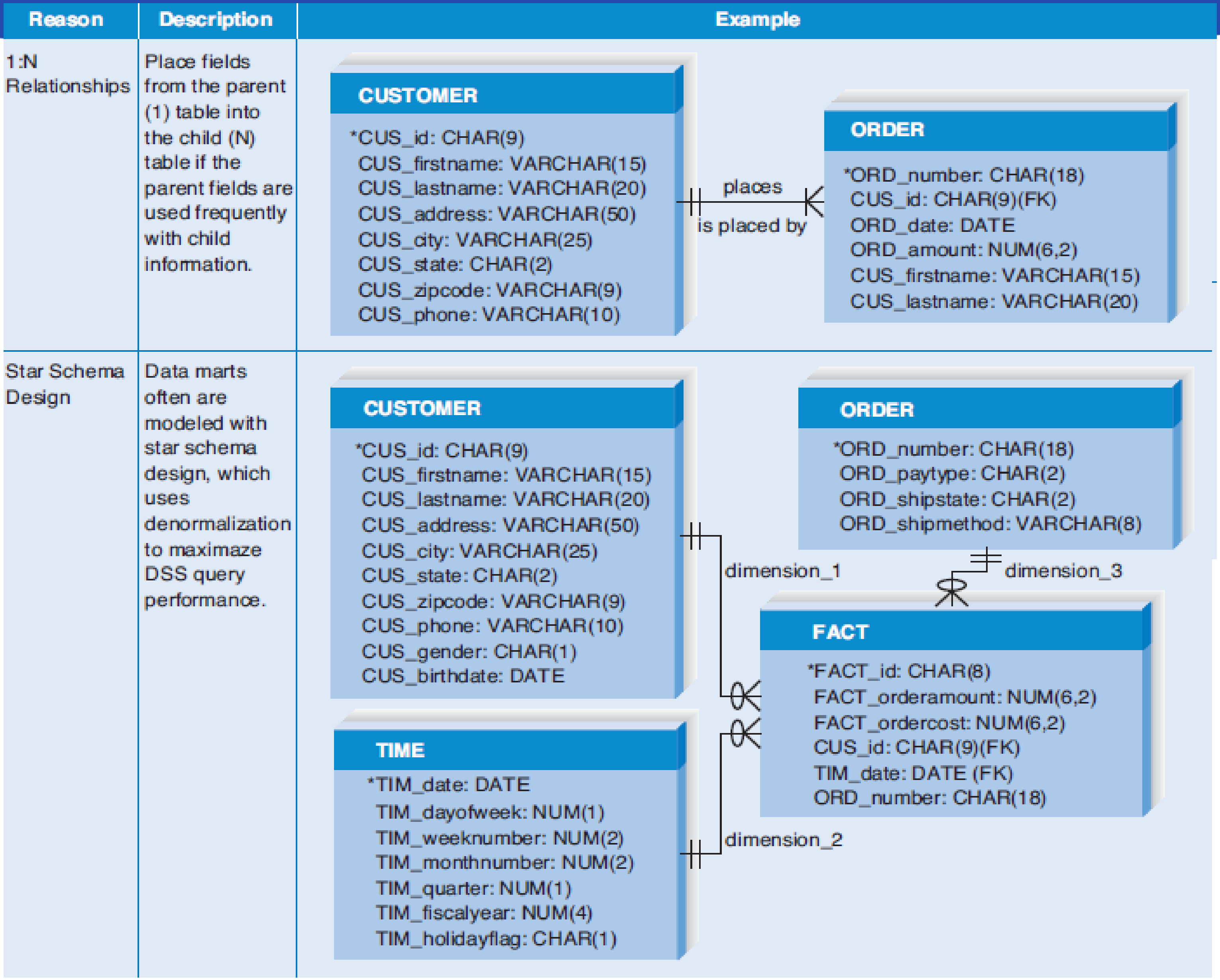
- Aspects:
  - for storage efficiency
  - for speed of access
- Optimizing for storage efficiency
  - Normalization is the best way to optimize data storage for efficiency.
- Optimizing for access speed
  - Denormalization

Optimizing data storage



Reason	Description	Example
Look-up Table	Include a code's description in the table using that code if the description is often used.	**ORDER** *ORD_number: CHAR(18) ORD_date: DATE ORD_amount: NUM(6,2) PAY_type: CHAR(2)(FK) PAY_description: VARCHAR(15) is paid by pays **PAYMENT_TYPE** *PAY_type: CHAR(2) PAY_description: VARCHAR(15)
1:1 Relationships	Combine tables if they are related 1:1 and if they usually are accessed together.	**ORDER** *ORD_number: CHAR(18) ORD_date: DATE ORD_amount: NUM(6,2) SHI_state: CHAR(2) SHI_method: CHAR(4) is sent by sends **SHIPMENT** *SHI_id: CHAR(9) ORD_num: CHAR(18) (FK) SHI_address: VARCHAR(50) SHI_city: VARCHAR(25) SHI_state: CHAR(2) SHI_zip: VARCHAR(9) SHI_method: CHAR(4)

Optimizing data storage



Reason	Description	Example
1:N Relationships	Place fields from the parent (1) table into the child (N) table if the parent fields are used frequently with child information.	CUSTOMER *CUS_id: CHAR(9) CUS_firstname: VARCHAR(15) CUS_lastname: VARCHAR(20) CUS_address: VARCHAR(50) CUS_city: VARCHAR(25) CUS_state: CHAR(2) CUS_zipcode: VARCHAR(9) CUS_phone: VARCHAR(10) places is placed by ORDER *ORD_number: CHAR(18) CUS_id: CHAR(9)(FK) ORD_date: DATE ORD_amount: NUM(6,2) CUS_firstname: VARCHAR(15) CUS_lastname: VARCHAR(20)
Star Schema Design	Data marts often are modeled with star schema design, which uses denormalization to maximaze DSS query performance.	CUSTOMER *CUS_id: CHAR(9) CUS_firstname: VARCHAR(15) CUS_lastname: VARCHAR(20) CUS_address: VARCHAR(50) CUS_city: VARCHAR(25) CUS_state: CHAR(2) CUS_zipcode: VARCHAR(9) CUS_phone: VARCHAR(10) CUS_gender: CHAR(1) CUS_birthdate: DATE ORDER *ORD_number: CHAR(18) ORD_paytype: CHAR(2) ORD_shipstate: CHAR(2) ORD_shipmethod: VARCHAR(8) dimension_1 dimension_3 FACT *FACT_id: CHAR(8) FACT_orderamount: NUM(6,2) FACT_ordercost: NUM(6,2) CUS_id: CHAR(9)(FK) TIM_date: DATE (FK) ORD_number: CHAR(18) dimension_2 TIME *TIM_date: DATE TIM_dayofweek: NUM(1) TIM_weeknumber: NUM(2) TIM_monthnumber: NUM(2) TIM_quarter: NUM(1) TIM_fiscalyear: NUM(4) TIM_holidayflag: CHAR(1)

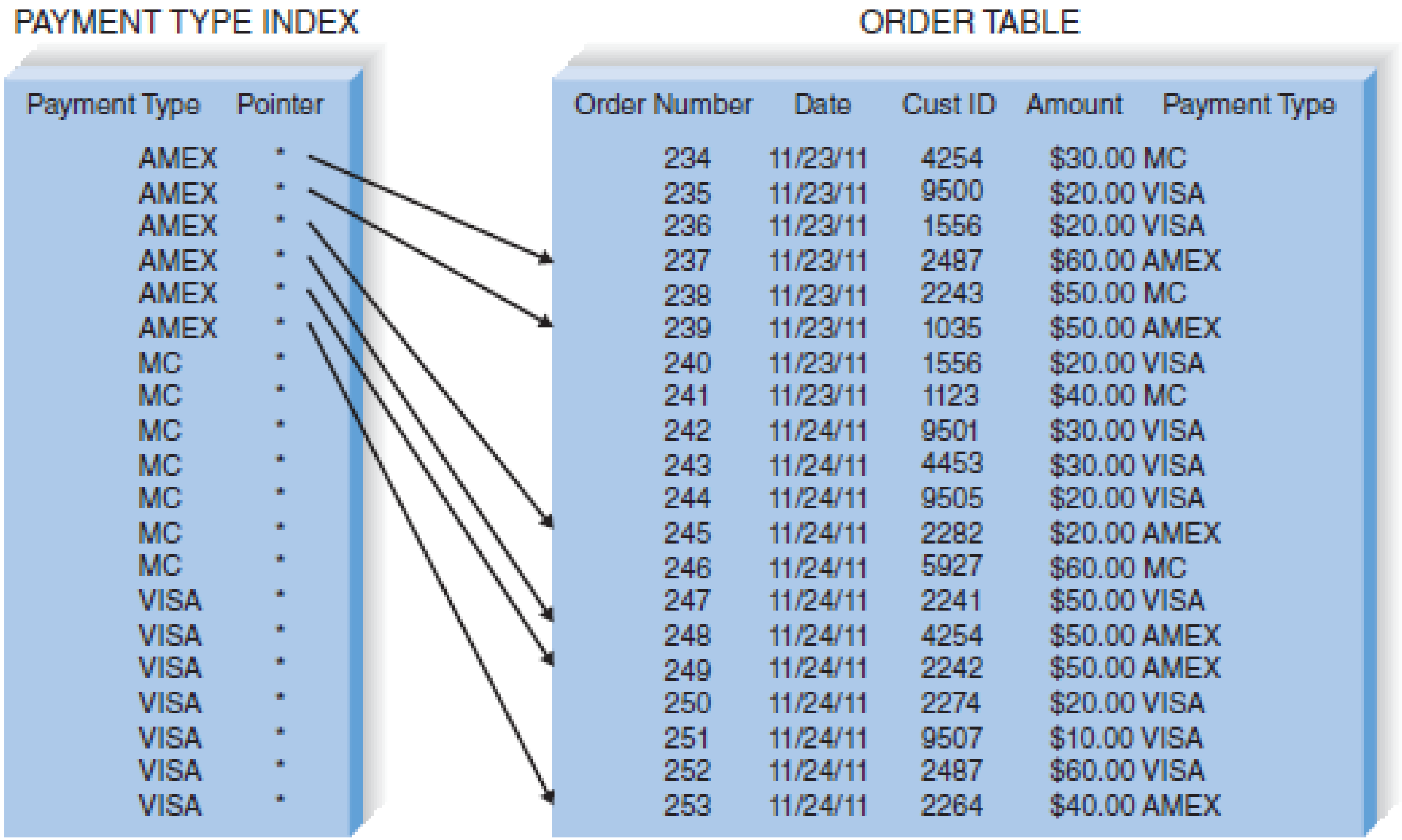
Optimizing data storage

- Clustering
  - Reduce the number of times that the storage medium must be accessed during a transaction.
  - Intrafile clustering
  - Interfile clustering
- Indexing

  Whenever you have performance problems, the first place to look is an index.

  - Storing index in memory

Optimizing data storage



PAYMENT TYPE INDEX

Payment Type	Pointer
AMEX	*
AMEX	*
AMEX	*
AMEX	*
AMEX	*
AMEX	*
MC	*
MC	*
MC	*
MC	*
MC	*
MC	*
MC	*
VISA	*
VISA	*
VISA	*
VISA	*
VISA	*
VISA	*
VISA	*

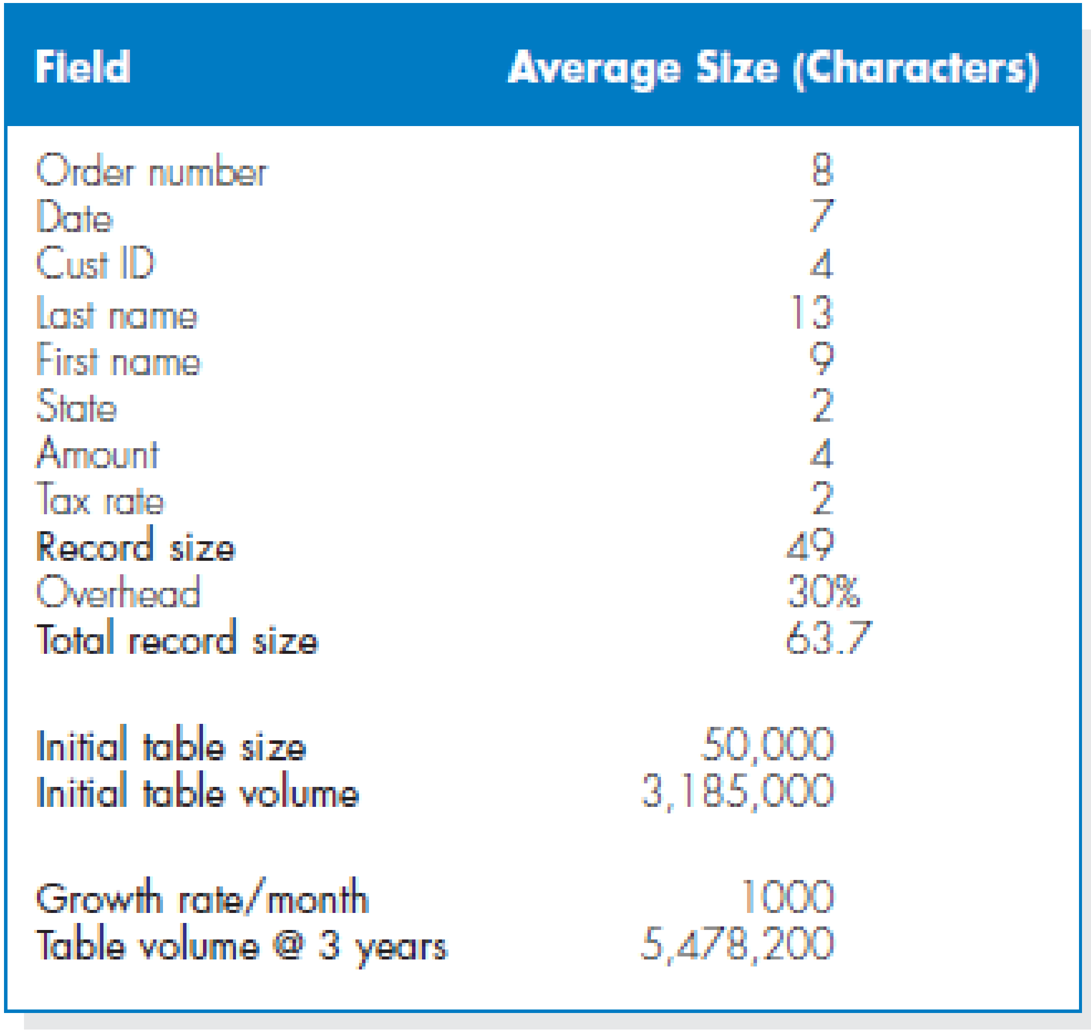
ORDER TABLE

Order Number	Date	Cust ID	Amount	Payment Type
234	11/23/11	4254	$30.00	MC
235	11/23/11	9500	$20.00	VISA
236	11/23/11	1556	$20.00	VISA
237	11/23/11	2487	$60.00	AMEX
238	11/23/11	2243	$50.00	MC
239	11/23/11	1035	$50.00	AMEX
240	11/23/11	1556	$20.00	VISA
241	11/23/11	1123	$40.00	MC
242	11/24/11	9501	$30.00	VISA
243	11/24/11	4453	$30.00	VISA
244	11/24/11	9505	$20.00	VISA
245	11/24/11	2282	$20.00	AMEX
246	11/24/11	5927	$60.00	MC
247	11/24/11	2241	$50.00	VISA
248	11/24/11	4254	$50.00	AMEX
249	11/24/11	2242	$50.00	AMEX
250	11/24/11	2274	$20.00	VISA
251	11/24/11	9507	$10.00	VISA
252	11/24/11	2487	$60.00	VISA
253	11/24/11	2264	$40.00	AMEX

Estimating Storage Size

- Volumetrics: estimating the amount of data that the hardware will need to support.
- One should know:
  - The initial size of data.
  - The expected growth rate over time.
  - Overhead
  - Hardware requirements
  - DBMS limitations
  - Hardware limitations

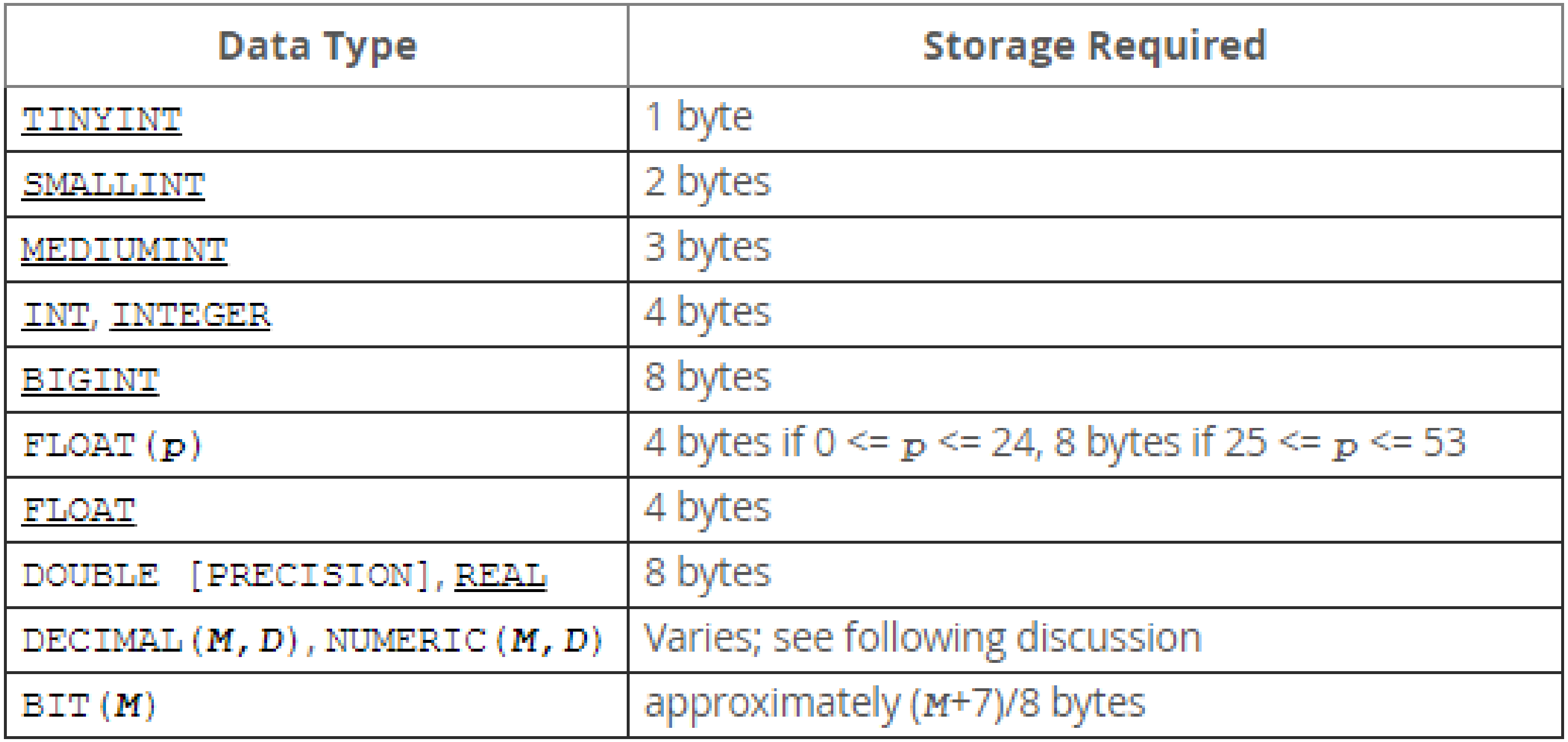
Estimating Storage Size



Field	Average Size (Characters)
Order number	8
Date	7
Cust ID	4
Last name	13
First name	9
State	2
Amount	4
Tax rate	2
Record size	49
Overhead	30%
Total record size	63.7
Initial table size	50,000
Initial table volume	3,185,000
Growth rate/month	1000
Table volume @ 3 years	5,478,200

Estimating Storage Size

- MySQL storage requirements

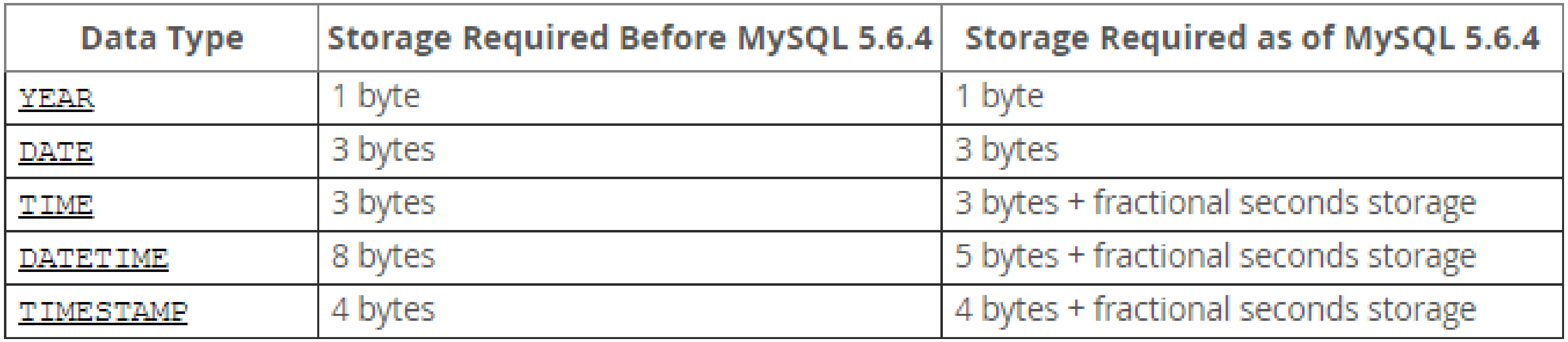


Data Type	Storage Required
TINYINT	1 byte
SMALLINT	2 bytes
MEDIUMINT	3 bytes
INT, INTEGER	4 bytes
BIGINT	8 bytes
FLOAT(*p*)	4 bytes if 0 <= *p* <= 24, 8 bytes if 25 <= *p* <= 53
FLOAT	4 bytes
DOUBLE [PRECISION], REAL	8 bytes
DECIMAL(*M*,*D*), NUMERIC(*M*,*D*)	Varies; see following discussion
BIT(*M*)	approximately (*M*+7)/8 bytes

https://dev.mysql.com/doc/refman/8.0/en/storage-requirements.html

Estimating Storage Size

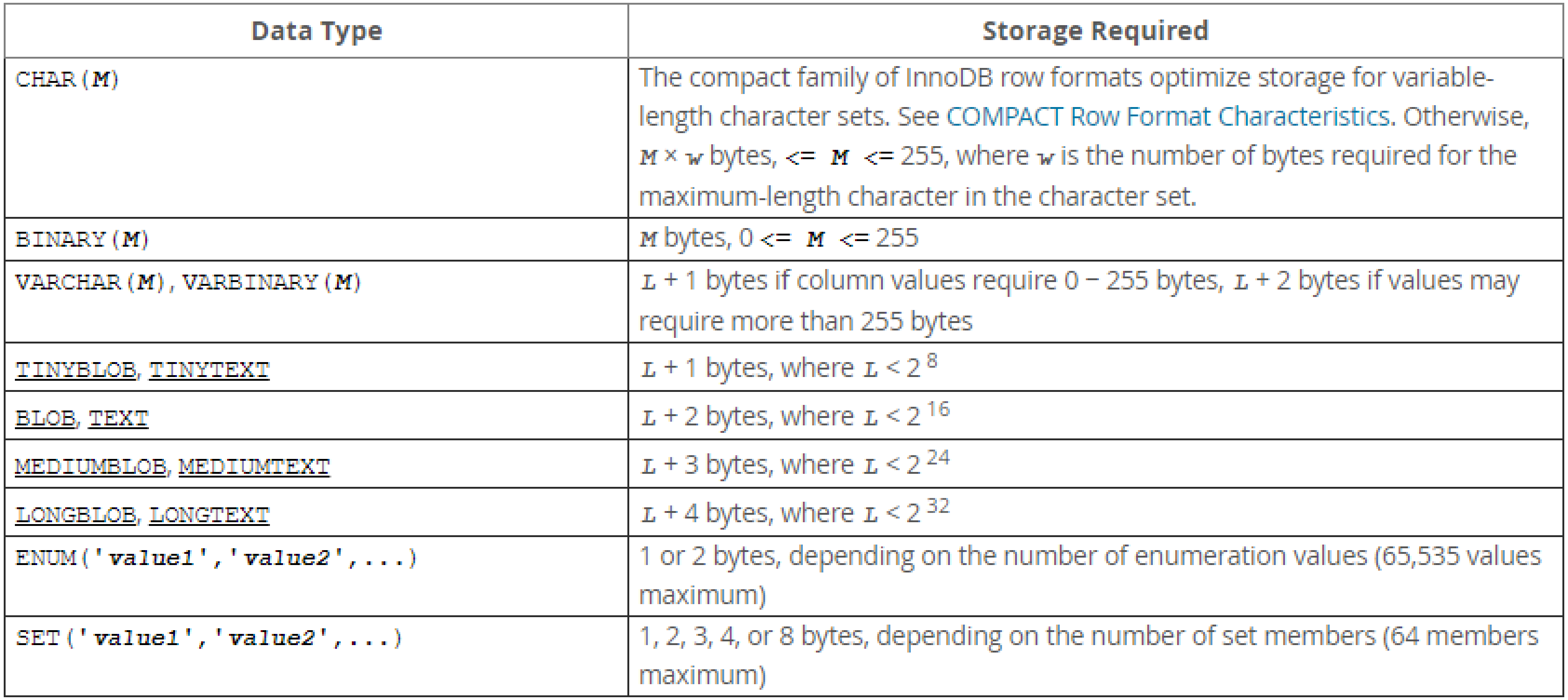
- MySQL storage requirements



Data Type	Storage Required Before MySQL 5.6.4	Storage Required as of MySQL 5.6.4
YEAR	1 byte	1 byte
DATE	3 bytes	3 bytes
TIME	3 bytes	3 bytes + fractional seconds storage
DATETIME	8 bytes	5 bytes + fractional seconds storage
TIMESTAMP	4 bytes	4 bytes + fractional seconds storage

https://dev.mysql.com/doc/refman/8.0/en/storage-requirements.html

Estimating Storage Size



Data Type	Storage Required
`CHAR(M)`	The compact family of InnoDB row formats optimize storage for variable-length character sets. See COMPACT Row Format Characteristics. Otherwise, $M \times w$ bytes, <= M <= 255, where w is the number of bytes required for the maximum-length character in the character set.
`BINARY(M)`	M bytes, 0 <= M <= 255
`VARCHAR(M)`, `VARBINARY(M)`	L + 1 bytes if column values require 0 − 255 bytes, L + 2 bytes if values may require more than 255 bytes
`TINYBLOB`, `TINYTEXT`	L + 1 bytes, where $L < 2^{8}$
`BLOB`, `TEXT`	L + 2 bytes, where $L < 2^{16}$
`MEDIUMBLOB`, `MEDIUMTEXT`	L + 3 bytes, where $L < 2^{24}$
`LONGBLOB`, `LONGTEXT`	L + 4 bytes, where $L < 2^{32}$
`ENUM('value1','value2',...)`	1 or 2 bytes, depending on the number of enumeration values (65,535 values maximum)
`SET('value1','value2',...)`	1, 2, 3, 4, or 8 bytes, depending on the number of set members (64 members maximum)

https://dev.mysql.com/doc/refman/8.0/en/storage-requirements.html

EFOP-3.4.3-16-2016-00009

A felsőfokú oktatás minőségének és hozzáférhetőségének együttes javítása a Pannon Egyetemen

THANKS FOR ATTENTION!

SZÉCHENYI 2020

Európai Unió
Európai Szociális Alap

MAGYARORSZÁG KORMÁNYA

BEFEKTETÉS A JÖVŐBE

EFOP-3.4.3-16-2016-00009

A felsőfokú oktatás minőségének és hozzáférhetőségének együttes javítása a Pannon Egyetemen

SYSTEM ANALYSIS AND SYSTEM DESIGN

11. IMPLEMENTATION &TRANSITION

Imre Budai

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE

Objectives

- Be familiar with the system construction process.
- Explain different types of tests and when to use them.
- Describe how to develop user documentation.
- Explain the system installation process.
- Describe the elements of a migration plan.
- Explain different types of conversion strategies and when to use them.
- Describe several techniques for managing change.
- Outline post installation processes.

Introduction

It must be remembered that there is nothing more difficult to plan, more doubtful of success, nor more dangerous to manage than the creation of a new system. For the initiator has the animosity of all who would profit by the preservation of the old institution and merely lukewarm defenders in those who would gain by the new.

—Machiavelli, The Prince, 1513

- Managing the change to a new is one of the most difficult tasks in any organization.
  - Business issues
  - Technical issues
  - People issues
- Leaving this planning to the last minute is a recipe for failure.

Introduction

- Developing and testing the system’s software
- Documentation
- New operating procedures
- Attention on managing the programming process
- designing a variety of tests that will be performed on the new system.
- Tests to verify that the system actually does what it was designed to do.
- Finalizing the system documentation
- Develop the user documentation.

Managing the programming process

- Assigning Programming Tasks
  - Grouping together modules that are related.
  - Assign these groups to programmers.
  - Assessment system:
    - Skill level
    - Experience
    - Other parameters
  - Eliminating deficiencies.
  - Need for:
    - Additional time in the project schedule
    - External resources

Managing the programming process

- Coordinating Activities
  - It can be done through both high-tech and low-tech means.
    - Weekly project meeting
  - Create and follow standards.
  - Separation of different roles (development, testing, production).
  - Git-like systems.
  - Change control
  - CASE tools are set up to track the status.

Managing the programming process

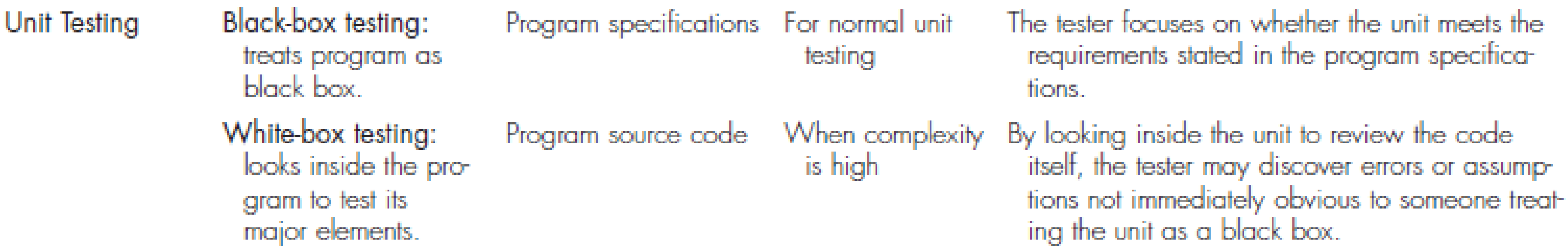
- Managing the Schedule
  - Initial planning.
  - Refined during the analysis and design.
  - Refined as the project progresses during construction.
  - Causes for schedule problems:
    - Scope creep – It can be very expensive.
    - Unnoticed day-by-day slippages.
  - Risk assessment.

Testing

- Testing and documentation aren’t fun.
- They receive less attention.
- Test Planning
  - Defines a series of tests that will be conducted.
  - It is impossible to test everything.
  - There are four general stages of tests:
    - Unit tests
    - integration tests
    - System tests
    - Acceptance tests

Testing

- Unit Tests
  - It focuses on one unit.
  - Done by the programmers.
  - The purpose of a unit test is to ensure that the module or program performs its function as defined in the program specification.
  - There are two approaches:
    - Black-box
    - White-box

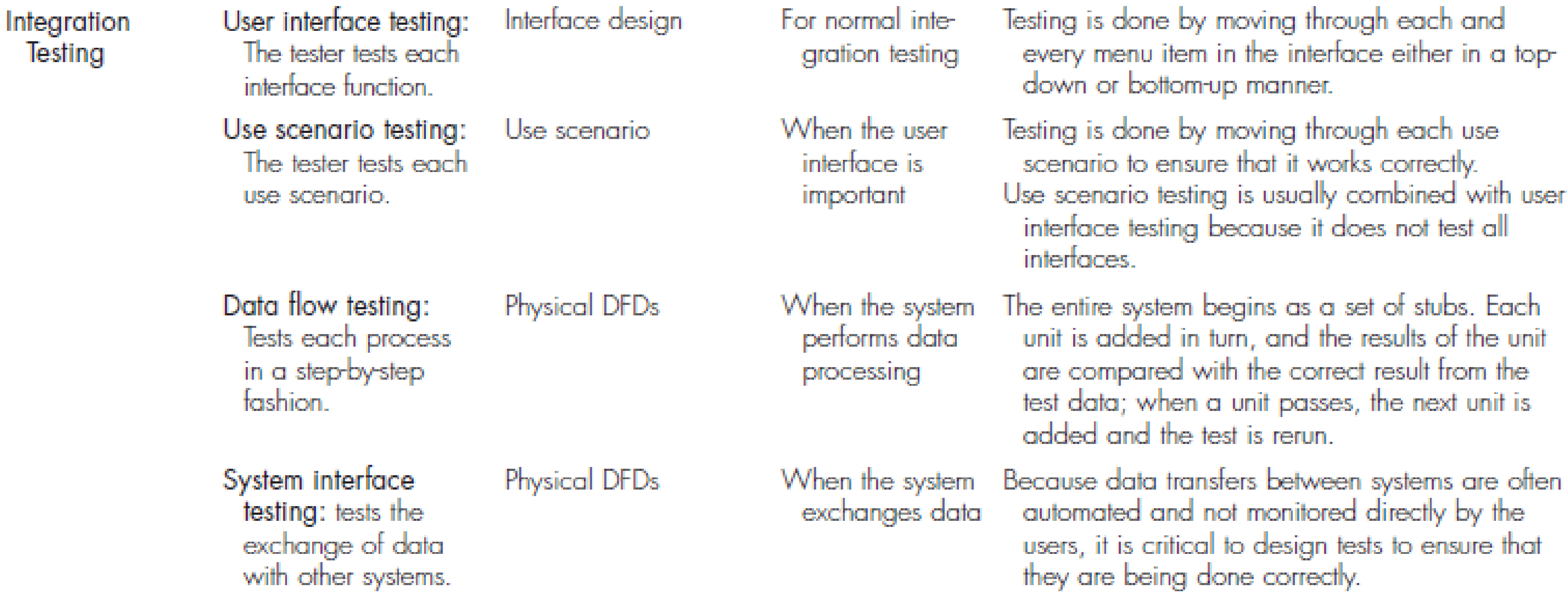


Unit Testing	Black-box testing: treats program as black box.	Program specifications	For normal unit testing	The tester focuses on whether the unit meets the requirements stated in the program specifications.
	White-box testing: looks inside the program to test its major elements.	Program source code	When complexity is high	By looking inside the unit to review the code itself, the tester may discover errors or assumptions not immediately obvious to someone treating the unit as a black box.

Testing

- Integration Tests
  - It assesses whether a set of modules or programs that must work together do so without error.
  - The focus now is on the:
    - Flow of control among modules.
    - Data exchanged among them.
  - It has the same general procedures as unit testing.
  - There are four approaches:
    - User interface testing.
    - Use scenario testing.
    - Data flow testing.
    - System interface testing.

Testing

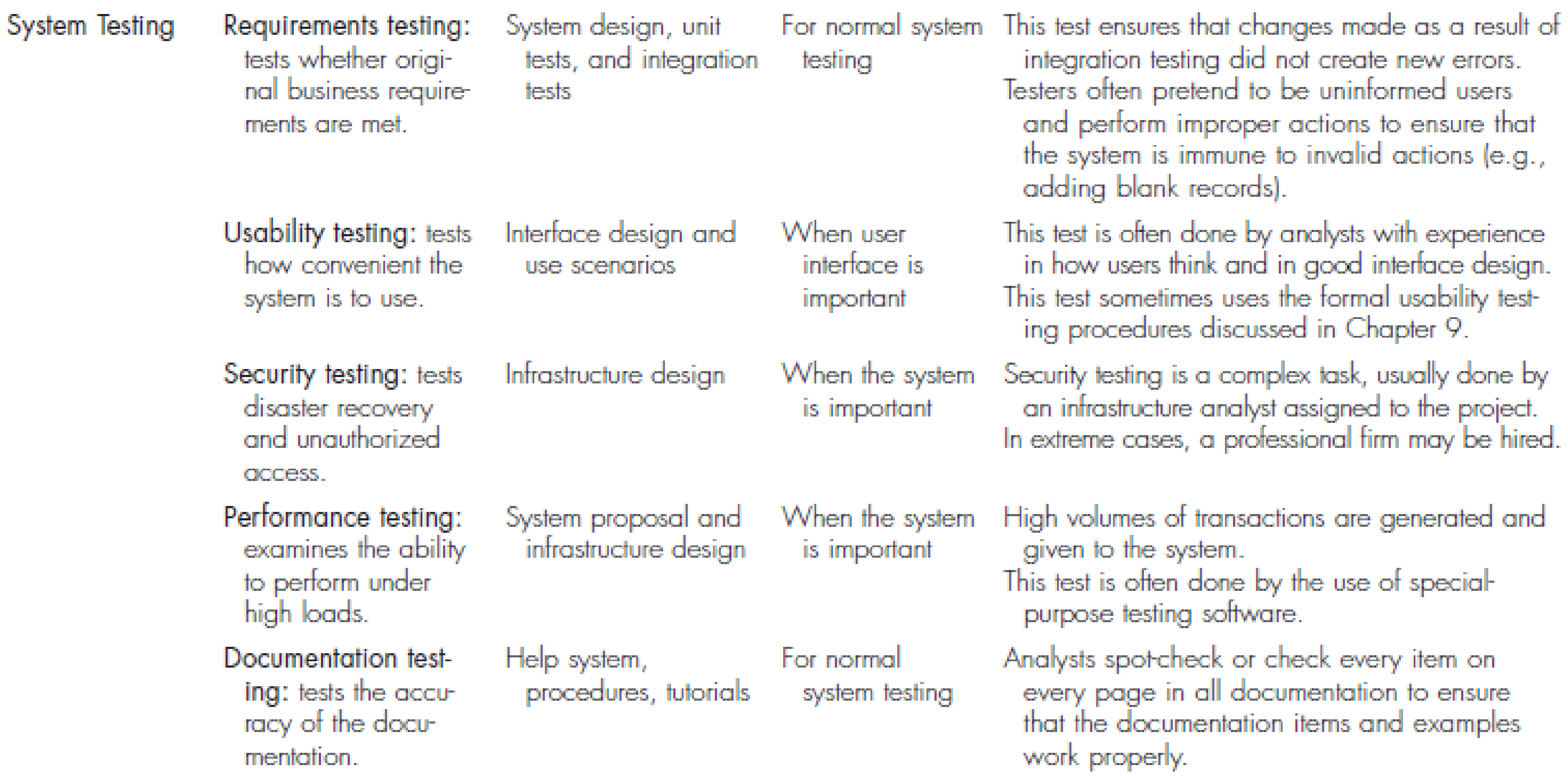


Integration Testing	User interface testing: The tester tests each interface function.	Interface design	For normal integration testing	Testing is done by moving through each and every menu item in the interface either in a top-down or bottom-up manner.
	Use scenario testing: The tester tests each use scenario.	Use scenario	When the user interface is important	Testing is done by moving through each use scenario to ensure that it works correctly. Use scenario testing is usually combined with user interface testing because it does not test all interfaces.
	Data flow testing: Tests each process in a step-by-step fashion.	Physical DFDs	When the system performs data processing	The entire system begins as a set of stubs. Each unit is added in turn, and the results of the unit are compared with the correct result from the test data; when a unit passes, the next unit is added and the test is rerun.
	System interface testing: tests the exchange of data with other systems.	Physical DFDs	When the system exchanges data	Because data transfers between systems are often automated and not monitored directly by the users, it is critical to design tests to ensure that they are being done correctly.

Testing

- System Tests
  - Usually conducted by the systems analysts.
  - To ensure that all modules and programs work together without error.
  - It has much broader in scope:
    - The system meets business requirements.
    - Usability.
    - Security.
    - Performance.
  - It also tests the system’s documentation.

Testing



System Testing	**Requirements testing:** tests whether original business requirements are met.	System design, unit tests, and integration tests	For normal system testing	This test ensures that changes made as a result of integration testing did not create new errors. Testers often pretend to be uninformed users and perform improper actions to ensure that the system is immune to invalid actions (e.g., adding blank records).
	Usability testing: tests how convenient the system is to use.	Interface design and use scenarios	When user interface is important	This test is often done by analysts with experience in how users think and in good interface design. This test sometimes uses the formal usability testing procedures discussed in Chapter 9.
	Security testing: tests disaster recovery and unauthorized access.	Infrastructure design	When the system is important	Security testing is a complex task, usually done by an infrastructure analyst assigned to the project. In extreme cases, a professional firm may be hired.
	Performance testing: examines the ability to perform under high loads.	System proposal and infrastructure design	When the system is important	High volumes of transactions are generated and given to the system. This test is often done by the use of special-purpose testing software.
	Documentation testing: tests the accuracy of the documentation.	Help system, procedures, tutorials	For normal system testing	Analysts spot-check or check every item on every page in all documentation to ensure that the documentation items and examples work properly.

Testing

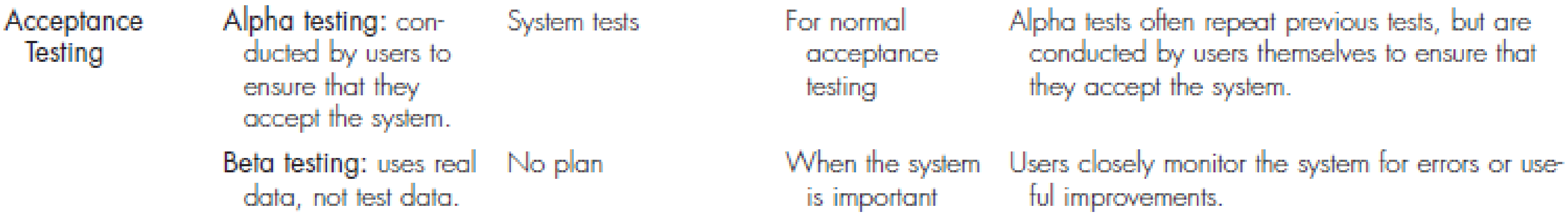
- Acceptance Tests
  - Primarily done by the users,
  - With support from the project team.
  - The goal is to:
    - Confirm that the system is complete.
    - Meets the business needs.
    - Acceptable to the users.

Testing

- Acceptance Tests
  - Two stages:
    - alpha testing

      Users test the system using made-up data
    - beta testing

      users begin to use the system with real data and carefully monitor the system for errors.



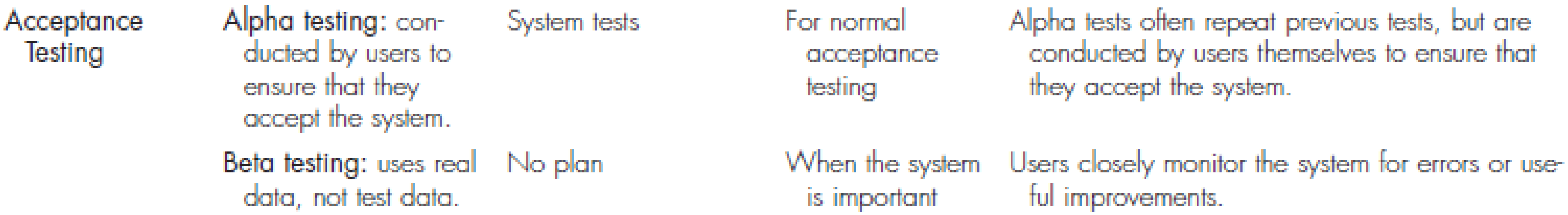
Acceptance Testing	**Alpha testing:** conducted by users to ensure that they accept the system.	System tests	For normal acceptance testing	Alpha tests often repeat previous tests, but are conducted by users themselves to ensure that they accept the system.
	Beta testing: uses real data, not test data.	No plan	When the system is important	Users closely monitor the system for errors or useful improvements.

Testing

- Acceptance Tests
  - Two stages:
    - alpha testing

      Users test the system using made-up data
    - beta testing

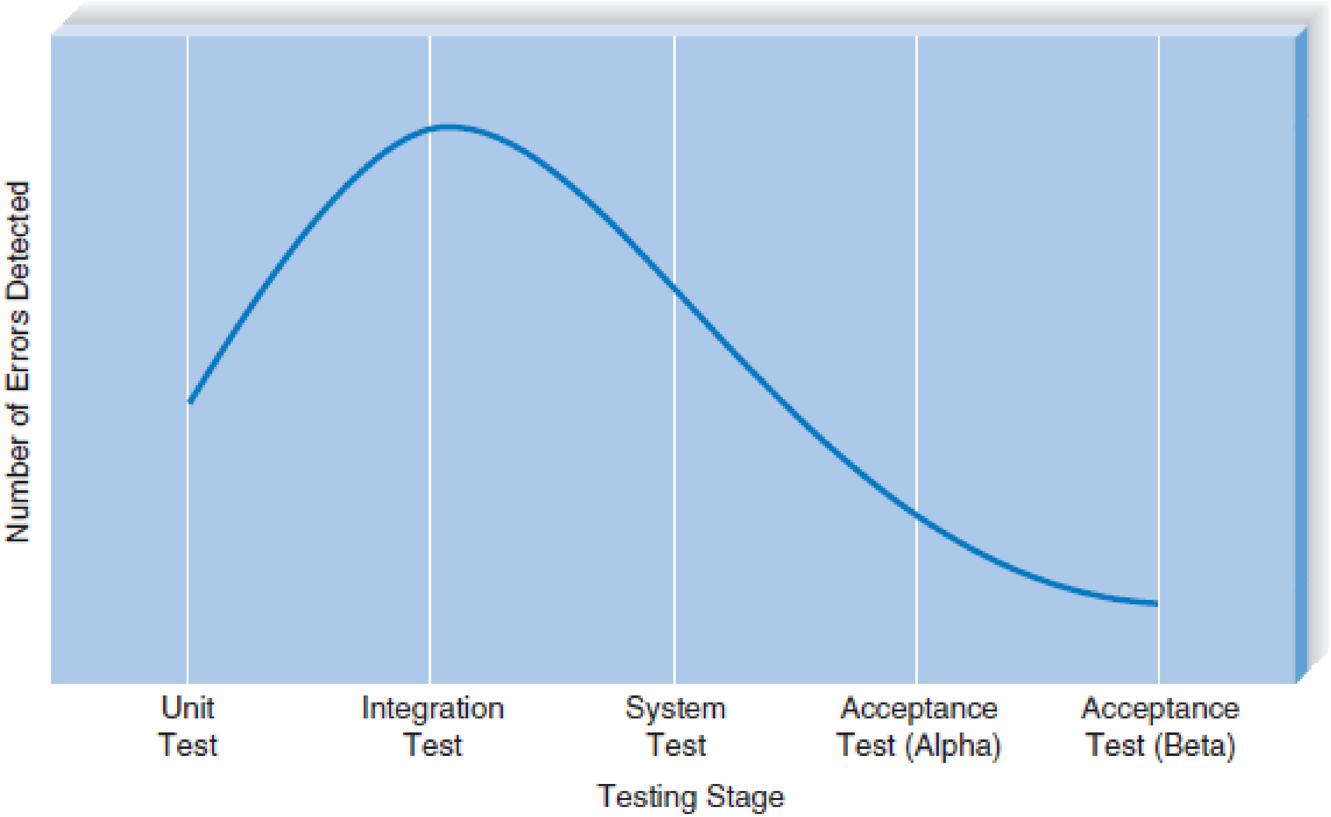
      users begin to use the system with real data and carefully monitor the system for errors.



Acceptance Testing	Alpha testing: conducted by users to ensure that they accept the system.	System tests	For normal acceptance testing	Alpha tests often repeat previous tests, but are conducted by users themselves to ensure that they accept the system.
	Beta testing: uses real data, not test data.	No plan	When the system is important	Users closely monitor the system for errors or useful improvements.

Testing

- Number of detected errors during testing phase.



Developing documentation

- There are two fundamentally different types of documentation:
  - System documentation

    Intended to help programmers and systems analysts understand the application software and enable them to build it or maintain it after the system is installed.
  - User documentation

    (user manuals, training manuals, and online help systems)

    Designed to help the user operate the system.
    - Time consuming process to write.
    - Can be started once the interface design and program specifications are complete.

Developing documentation

- User documentation
  - Key strengths of online documentation:
    - Searching for information.
    - The same information can be presented several times in many different formats.
    - Enabling the user to interact with the documentation.
    - Less expensive to distribute and keep up to date.

Developing documentation

- Types of Documentation
  - There are three fundamentally different types of user documentation:
    - Reference documents.

      Designed to be used when the user needs to learn how to perform a specific function.
    - Procedures manuals.

      Describe how to perform business tasks.
    - Tutorials.

      Teach people how to use major components of the system.

Making the transition

- Change is a three-step process:
  - Unfreeze the existing habits and norms (the as-is system)
    - Aware of the new system.
    - Participation in an analysis and design.
  - Move or transition, from the old system to the new.
    - Conversion strategy.
    - Switch from the old to the new system.
    - Plans to handle potential business disruption.
    - Helping the people who are affected by the new system
  - Refreeze the new system.
    - Becoming the standard way of performing the business functions it supports.

Making the transition

- Post implementation activities:
  - system support

    Providing help desk and telephone support for users.
  - System maintenance

    Fixing bugs and improving the system after it has been installed.
  - project assessment

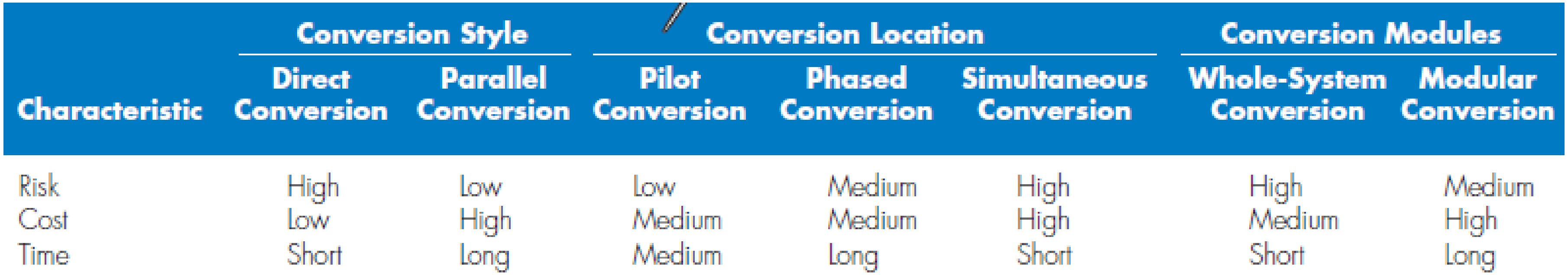
    Process of evaluating the project to identify what went well and what could be improved for the next system development project.

Migration plan

- Preparing:
  - Business.
  - Technology.
  - People.
- Selecting the conversion strategy.
  - Conversion style.
  - Conversion locations.
  - Conversion modules.
- Evaluating the strategy choices

Migration plan

- Main characteristics



	Conversion Style		Conversion Location			Conversion Modules	
Characteristic	**Direct Conversion**	**Parallel Conversion**	**Pilot Conversion**	**Phased Conversion**	**Simultaneous Conversion**	**Whole-System Conversion**	**Modular Conversion**
Risk	High	Low	Low	Medium	High	High	Medium
Cost	Low	High	Medium	Medium	High	Medium	High
Time	Short	Long	Medium	Long	Short	Short	Long

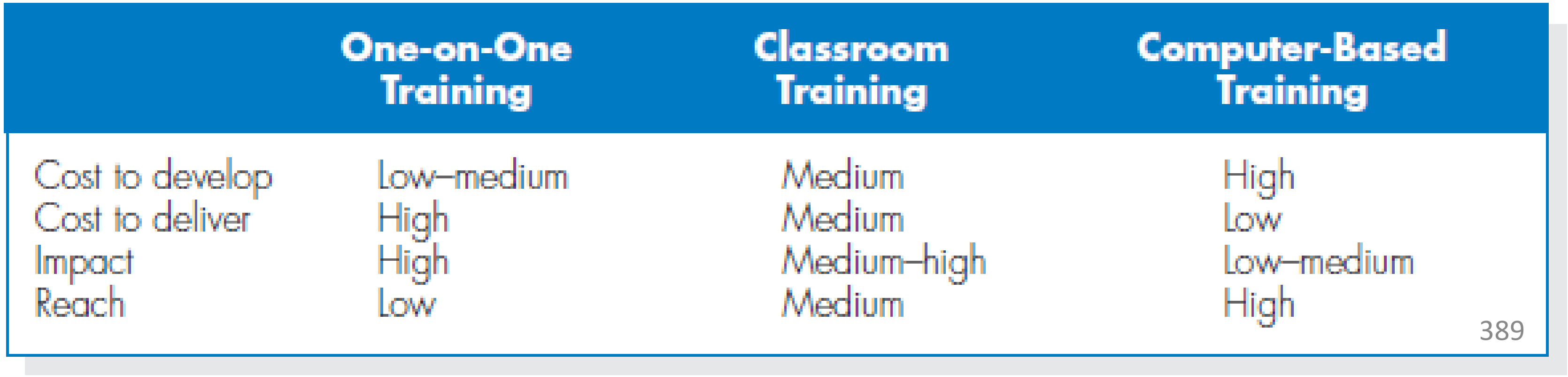
Migration plan

- Preparing a business contingency plan.
- Preparing the technology
  - Install the hardware.
  - Install the software.
  - Convert the data.
- Preparing people for the new system.

  People need help to adopt and adapt to the new system.
- Understanding resistance to change.
- Revising management policies

Migration plan

- Assessing costs and benefits.
- Motivating adoption
    - Informal
    - Political
- Enabling adoption: Training
    - What to train
    - How to train



	One-on-One Training	Classroom Training	Computer-Based Training
Cost to develop	Low–medium	Medium	High
Cost to deliver	High	Medium	Low
Impact	High	Medium–high	Low–medium
Reach	Low	Medium	High

EFOP-3.4.3-16-2016-00009

A felsőfokú oktatás minőségének és hozzáférhetőségének együttes javítása a Pannon Egyetemen

THANKS FOR ATTENTION!

SZÉCHENYI 2020

MAGYARORSZÁG KORMÁNYA

Európai Unió
Európai Szociális Alap

BEFEKTETÉS A JÖVŐBE