

Requirement Analysis & Specification

TOPIC # 8

Chapter 10,11,12 & 13 – Karl Wieggers

Chapter 9 - Reference

What is Requirements Analysis

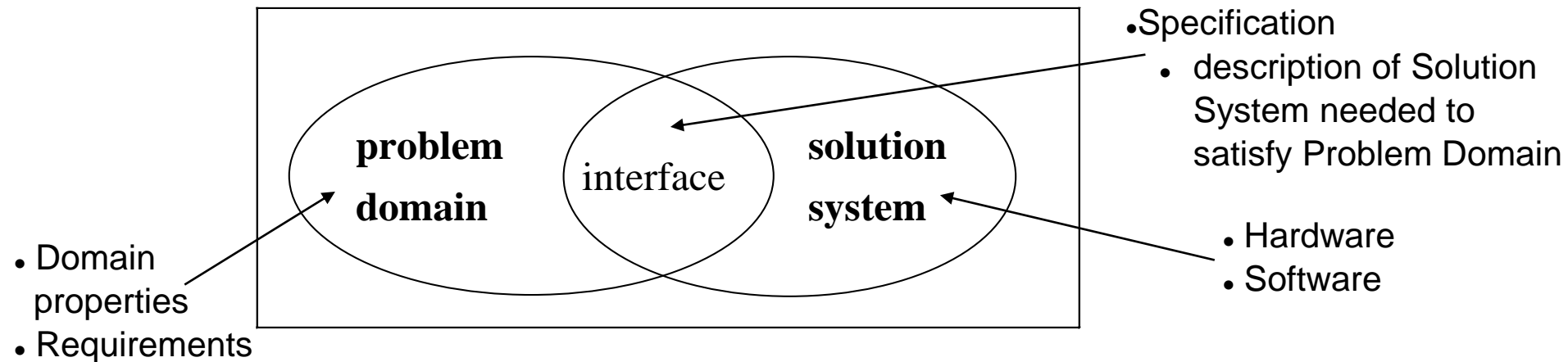
The process of studying and analyzing the customer and the user needs to arrive at a definition of the problem domain and system requirements

Objectives

- Discover the boundaries of the new system (or software) and how it must interact with its environment within the new problem domain
- Detect and resolve conflicts between (user) requirements
- Negotiate priorities of stakeholders
- Prioritize and triage requirements
- Elaborate system requirements, defined in the requirement specification document, such that managers can give realistic project estimates and developers can design, implement, and test
- Classify requirements information into various categories and allocate requirements to sub-systems
- Evaluate requirements for desirable qualities

What is Requirements Specification?

- The invention and definition of the behavior of a new system (solution domain) such that it will produce the required effects in the problem domain
- Start from a knowledge of the problem domain and the required effects determined by elicitation and analysis
- Generally involves **modeling**
- Results in the specification document
 - Precise expression of requirements, may include models



Software Requirement Specification (SRS)

- The software requirements specification goes by many names in various organizations, although organizations do not use these terms in the same way.
- It is sometimes called a *business requirements document* (BRD), *functional specification*, *product specification*, *system specification*, or simply *requirements document*. Because “software requirements specification” is an industry-standard term.
- The SRS states the functions and capabilities that a software system must provide, its characteristics, and the constraints that it must respect.

SRS Audience

- Customers, the marketing department, and sales staff need to know what product they can expect to be delivered.
- Project managers base their estimates of schedule, effort, and resources on the requirements.
- Software development teams need to know what to build.
- Testers use it to develop requirements-based tests, test plans, and test procedures.
- Maintenance and support staff use it to understand what each part of the product is supposed to do.
- Documentation writers base user manuals and help screens on the SRS and the user interface design.
- Training personnel use the SRS and user documentation to develop educational materials.
- Legal staff ensures that the requirements comply with applicable laws and regulations.
- Subcontractors base their work on—and can be legally held to—the specified requirements.

Labeling Requirements

- Every requirement needs a unique and persistent identifier.
- This allows one to refer to specific requirements in a change request, modification history, cross-reference, or requirements traceability matrix.
- *It also enables reusing the requirements in multiple projects.
- Simple numbered or bulleted lists aren't adequate for these purposes.
- Some ways to label requirements are:
- **Sequence Number:**
 - The simplest approach gives every requirement a unique sequence number, such as UC-9 or FR-26 (FR Functional requirement).

Labeling Requirements

- **Hierarchical Numbering:**
- In the most commonly used convention, if the functional requirements appear in section 3.2 of your SRS, they will all have labels that begin with 3.2.
- More digits indicate a more detailed, lower-level requirement, so you know that 3.2.4.3 is a child requirement of 3.2.4.
- An improvement over hierarchical numbering is to number the major sections of the requirements hierarchically and then identify individual functional requirements in each section with a short text code followed by a sequence number. For example, the SRS might contain “Section 3.5—Editor Functions,” and the requirements in that section could be labeled ED-1, ED-2, and so forth.

Labeling Requirements

- **Hierarchical Textual Tags:**

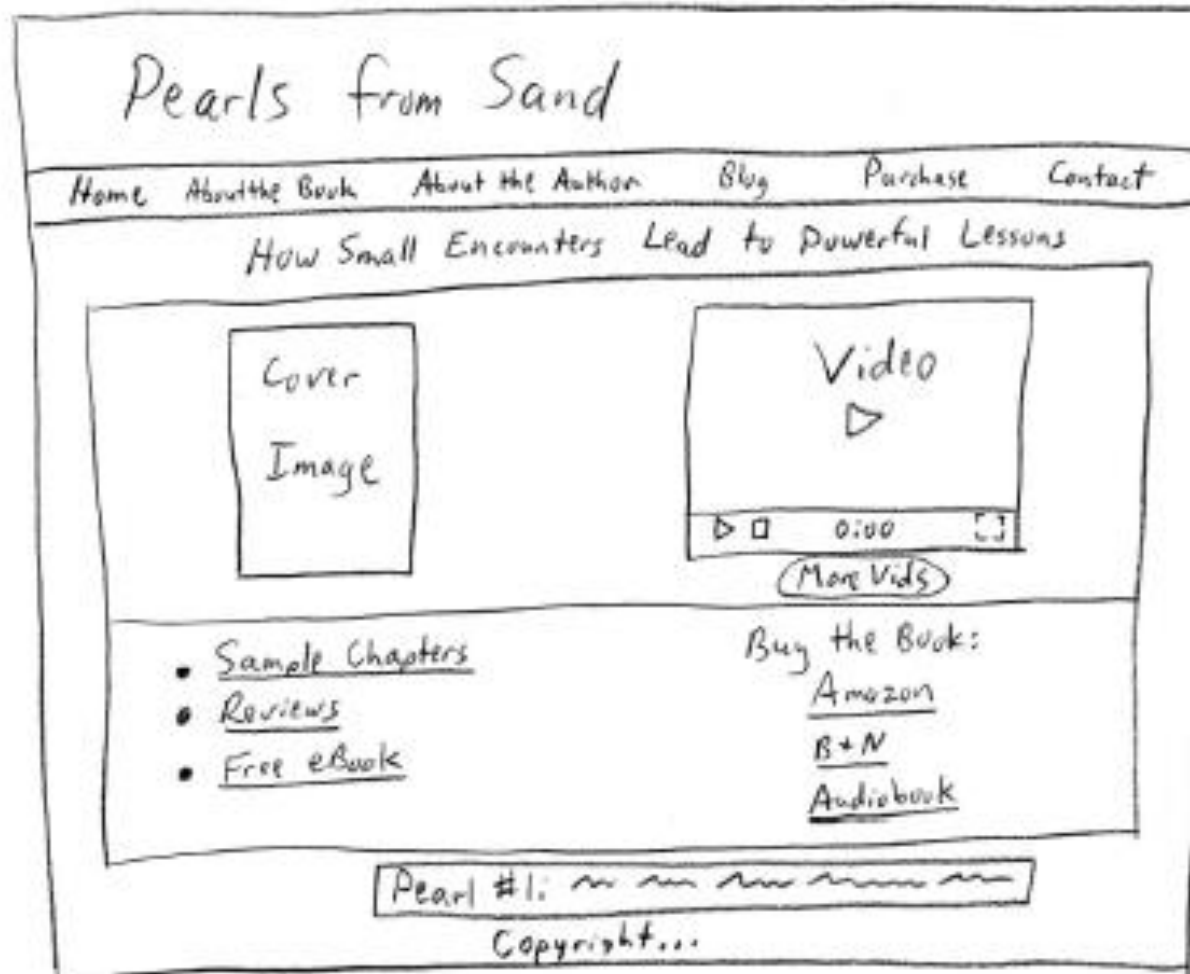
Product: Ordering products from the website

.Cart	The website shall use a shopping cart to contain products a Customer selects to purchase.
.Discount	The shopping cart shall provide one discount code field. Each discount code provides either a specific discount percentage or a fixed dollar discount amount on specific items in the cart.
.Error	If the Customer enters an invalid discount code, the website shall display an error message.
.Shipping	The shopping cart shall add a shipping charge if a Customer orders a physical product that must be mailed.

User Interfaces & the SRS

- Incorporating user interface designs in the SRS has both benefits and drawbacks.
- On the plus side, exploring possible user interfaces with paper prototypes, working mock-ups, wireframes, or simulation tools makes the requirements tangible to both users and developers.
- These are powerful techniques for eliciting and validating requirements.
- If you really do want to implement certain functionality with specific UI controls and screen layouts, it's both appropriate and important to include that information in the SRS as design constraints. Design constraints restrict the choices available to the user interface designer.

Example – Wire Frame & User Interface sketch



Example of a user interface "sketch" suitable for inclusion in a requirements document.

Proposed Template for SRS

1. Introduction
 - 1.1 Purpose
 - 1.2 Document conventions
 - 1.3 Project scope
 - 1.4 References
2. Overall description
 - 2.1 Product perspective
 - 2.2 User classes and characteristics
 - 2.3 Operating environment
 - 2.4 Design and implementation constraints
 - 2.5 Assumptions and dependencies
3. System features
 - 3.x System feature X
 - 3.x.1 Description
 - 3.x.2 Functional requirements
4. Data requirements
 - 4.1 Logical data model
 - 4.2 Data dictionary
 - 4.3 Reports
 - 4.4 Data acquisition, integrity, retention, and disposal
5. External interface requirements
 - 5.1 User interfaces
 - 5.2 Software interfaces
 - 5.3 Hardware interfaces
 - 5.4 Communications interfaces
6. Quality attributes
 - 6.1 Usability
 - 6.2 Performance
 - 6.3 Security
 - 6.4 Safety
 - 6.x [others]
7. Internationalization and localization requirements
8. Other requirements
- Appendix A: Glossary
- Appendix B: Analysis models

FROM THE VOICE OF CUSTOMER TO MODELS

Type of word	Examples	Analysis model components
Noun	People, organizations, software systems, data elements, or objects that exist	<ul style="list-style-type: none">■ External entities, data stores, or data flow (DFD)■ Actors (use case diagram)■ Entities or their attributes (ERD)■ Lanes (swimlane diagram)■ Objects with states (STD)
Verb	Actions, things a user or system can do, or events that can take place	<ul style="list-style-type: none">■ Processes (DFD)■ Process steps (swimlane diagram)■ Use cases (use case diagram)■ Relationships (ERD)■ Transitions (STD)■ Activities (activity diagram)■ Events (event-response table)
Conditional	Conditional logic statements, such as if/then	<ul style="list-style-type: none">■ Decisions (decision tree, decision table, or activity diagram)■ Branching (swimlane diagram or activity diagram)

Example: Chemical Tracking System

- For understanding important nouns are bold, verbs being italic and conditional statements are bold, italic.

A **chemist** or a member of the **chemical stockroom staff** can *place* a request for one or more **chemicals** *if the user is an authorized requester*. The request can be *fulfilled* either by *delivering* a **container** of the chemical that is already in the chemical stockroom's **inventory** or by placing an **order** for a new container of the chemical with an outside **vendor**. *If the chemical is hazardous*, the chemical can be delivered only *if the user is trained*. The **person** placing the request must be able to *search* **vendor catalogs** online for specific chemicals while *preparing* his request. The system needs to *track* the **status** of every chemical request from the time it is prepared until the request is either fulfilled or *canceled*. It also needs to track the **history** of every chemical container from the time it is *received* at the **company** until it is fully *consumed* or *disposed* of.

Selecting Right Representations

Information depicted	Representation techniques
System external interfaces	<ul style="list-style-type: none"> ■ The <i>context diagram</i> and <i>use case diagram</i> identify objects outside the system that connect to it. The <i>context diagram</i> and <i>data flow diagrams</i> illustrate the system inputs and outputs at a high level of abstraction. The <i>ecosystem map</i> identifies possible systems that interact, but includes some that do not interface directly as well. <i>Swimlane diagrams</i> show what happens in the interactions between systems. ■ External interface details can be recorded in input and output <i>file formats</i> or <i>report layouts</i>. Products that include both software and hardware components often have <i>interface specifications</i> with data attribute definitions, perhaps in the form of an application programming interface or specific input and output signals for a hardware device.
Business process flow	<ul style="list-style-type: none"> ■ A top-level <i>data flow diagram</i> represents how a business process handles data at a high level of abstraction. <i>Swimlane diagrams</i> show the roles that participate in executing the various steps in a business process flow. ■ Refined levels of <i>data flow diagrams</i> or <i>swimlane diagrams</i> can represent business process flows in considerable detail. Similarly, <i>flowcharts</i> and <i>activity diagrams</i> can be used at either high or low levels of abstraction, although most commonly they are used to define the details of a process.
Data definitions and data object relationships	<ul style="list-style-type: none"> ■ The <i>entity-relationship diagram</i> shows the logical relationships between data objects (entities). <i>Class diagrams</i> show the logical connections between object classes and the data associated with them. ■ The <i>data dictionary</i> contains detailed definitions of data structures and individual data items. Complex data objects are progressively broken down into their constituent data elements.
System and object states	<ul style="list-style-type: none"> ■ <i>State-transition diagrams</i> and <i>state tables</i> represent a high-abstraction view of the possible states of a system or object and the changes between states that can take place under certain circumstances. These models are helpful when multiple use cases can manipulate (and change the state of) certain objects. ■ Some analysts create an <i>event-response table</i> as a scoping tool, identifying external events that help define the product's scope boundary. You can also specify individual functional requirements with an event-response table by detailing how the system should behave in response to each combination of external event and system state. ■ <i>Functional requirements</i> provide the details that describe exactly what user and system behaviors lead to status changes.
Complex logic	<ul style="list-style-type: none"> ■ A <i>decision tree</i> shows the possible outcomes from a set of related decisions or conditions. A <i>decision table</i> identifies the unique functional requirements associated with the various combinations of true and false outcomes for a series of decisions or conditions.

Selecting Right Representations

Information depicted	Representation techniques
User interfaces	<ul style="list-style-type: none">■ The <i>dialog map</i> provides a high-level view of a proposed or actual user interface, showing the various display elements and possible navigation pathways between them.■ <i>Storyboards</i> and <i>low-fidelity prototypes</i> flesh out the dialog map by showing what each screen will contain without depicting precise details. <i>Display-action-response models</i> describe the display and behavior requirements of each screen.■ <i>Detailed screen layouts</i> and <i>high-fidelity prototypes</i> show exactly how the display elements will look. <i>Data field definitions</i> and <i>user interface control descriptions</i> provide additional detail.
User task descriptions	<ul style="list-style-type: none">■ <i>User stories</i>, <i>scenarios</i>, and <i>use case specifications</i> describe user tasks in various levels of detail.■ <i>Swimlane diagrams</i> illustrate the business process or interplay between multiple actors and the system. <i>Flowcharts</i> and <i>activity diagrams</i> visually depict the flow of the use case dialog and branches into alternative flows and exceptions.■ <i>Functional requirements</i> provide detailed descriptions of how the system and user will interact to achieve valuable outcomes. <i>Test cases</i> provide an alternative low-abstraction view, describing exactly what system behavior to expect under specific conditions of inputs, system state, and actions.
Nonfunctional requirements (quality attributes, constraints)	<ul style="list-style-type: none">■ Quality attributes and constraints are usually written in the form of <i>natural language text</i>, but that often results in a lack of precision and completeness.

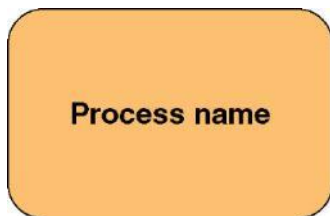
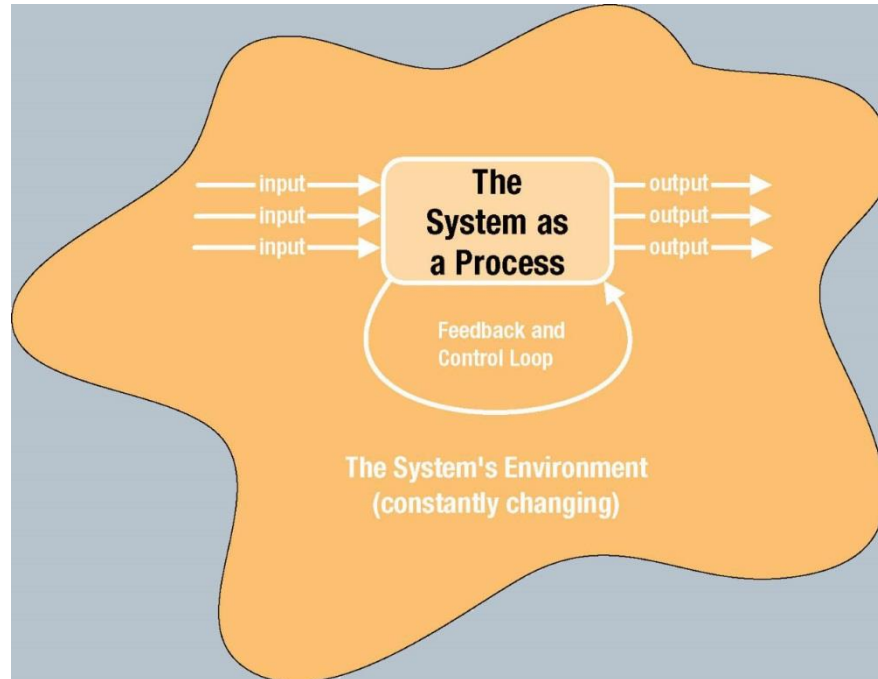
Data Flow diagrams

Data flow diagram (DFD) – a process model used to depict the flow of data through a system and the work or processing performed by the system. Synonyms are bubble chart, transformation graph, and process model.

DFDs have become a popular tool for business process redesign.

Process Concepts

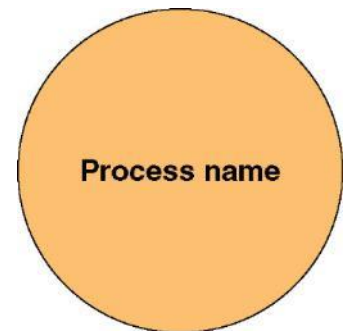
Process – work performed by a system in response to incoming data flows or conditions. A synonym is *transform*.



Gane & Sarson shape;
used throughout
this book



SSADM/IDEF0 shape
Process Symbols



DeMarco/Yourdon shape

External Agents

External agent – an outside person, organization unit, system, or organization that interacts with a system. Also called an *external entity*.

- External agents define the “boundary” or scope of a system being modeled.
- As scope changes, external agents can become processes, and vice versa.
- Almost always one of the following:
 - Office, department, division.
 - An external organization or agency.
 - Another business or another information system.
 - One of your system’s end-users or managers
- Named with descriptive, singular noun



Gane and Sarson shape



DeMarco/Yourdon shape

Data Stores

Data store – stored data intended for later use. Synonyms are *file* and *database*.

- Frequently implemented as a file or database.
- A data store is “data at rest” compared to a data flow that is “data in motion.”

– Almost always one of the following:

- Persons (or groups of persons)
- Places
- Objects
- Events (about which data is captured)
- Concepts (about which data is important)

– Data stores depicted on a DFD store all instances of data entities (depicted on an ERD)

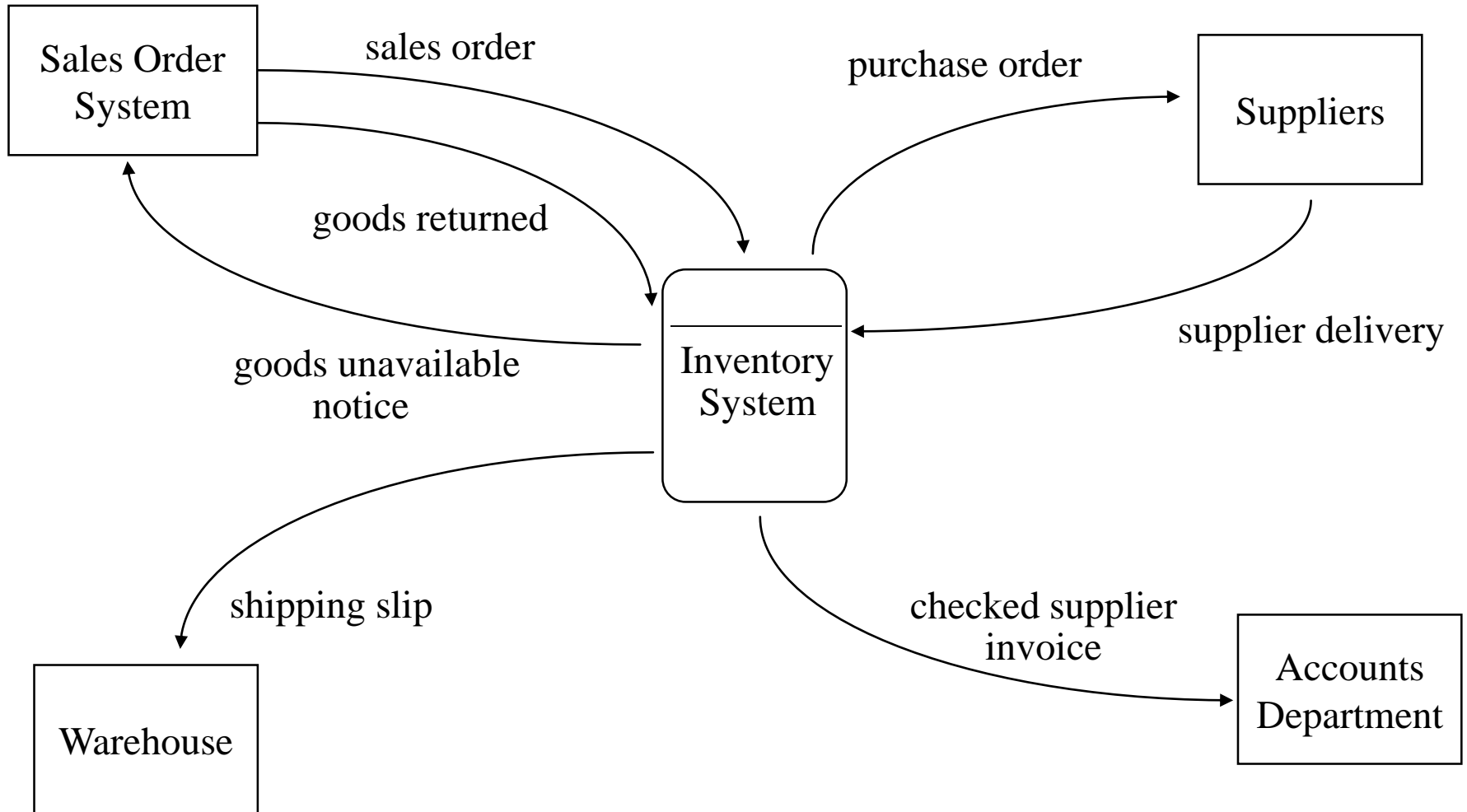


Gane and Sarson shape

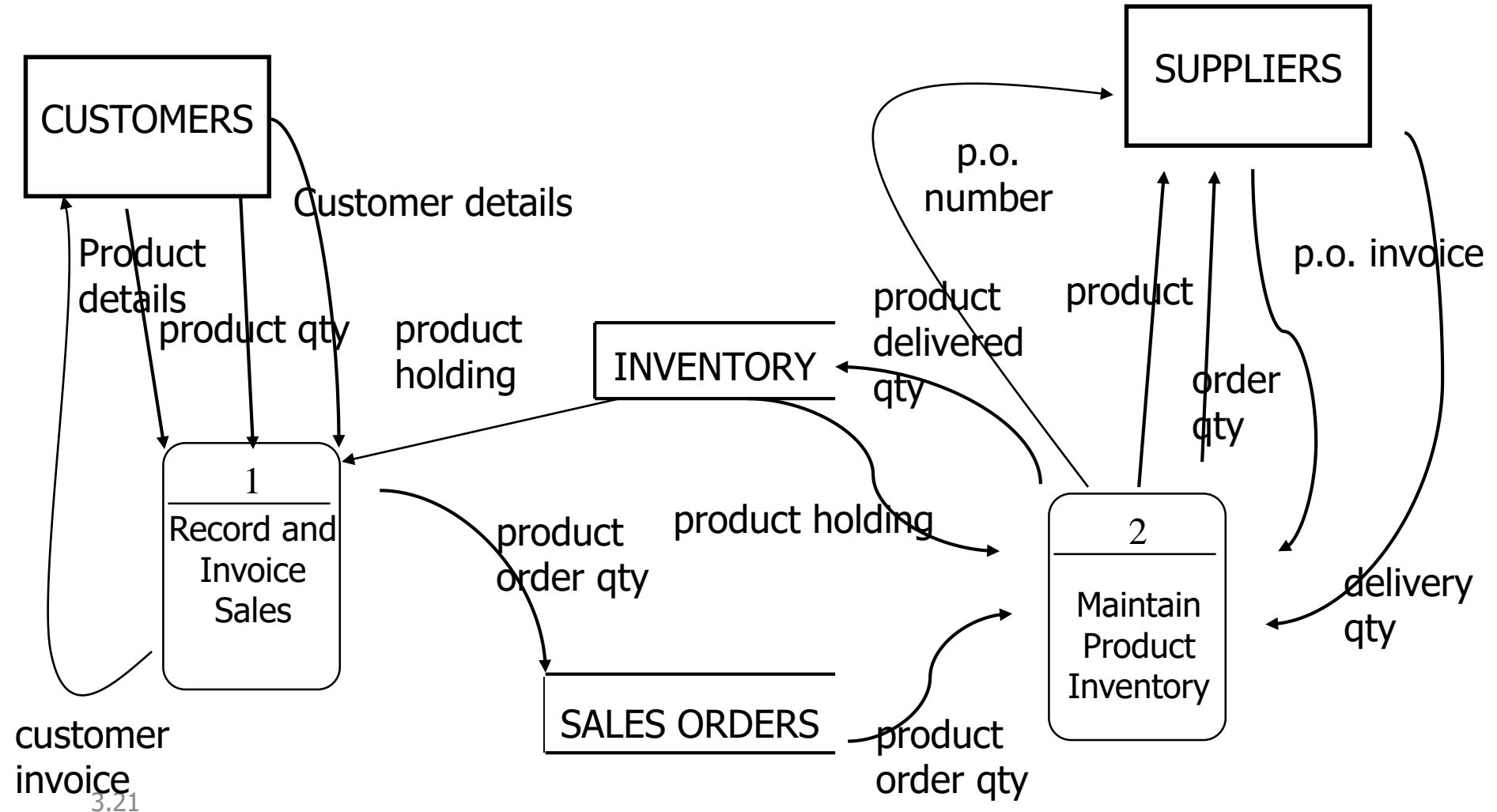
Data Store

DeMarco/Yourdon shape

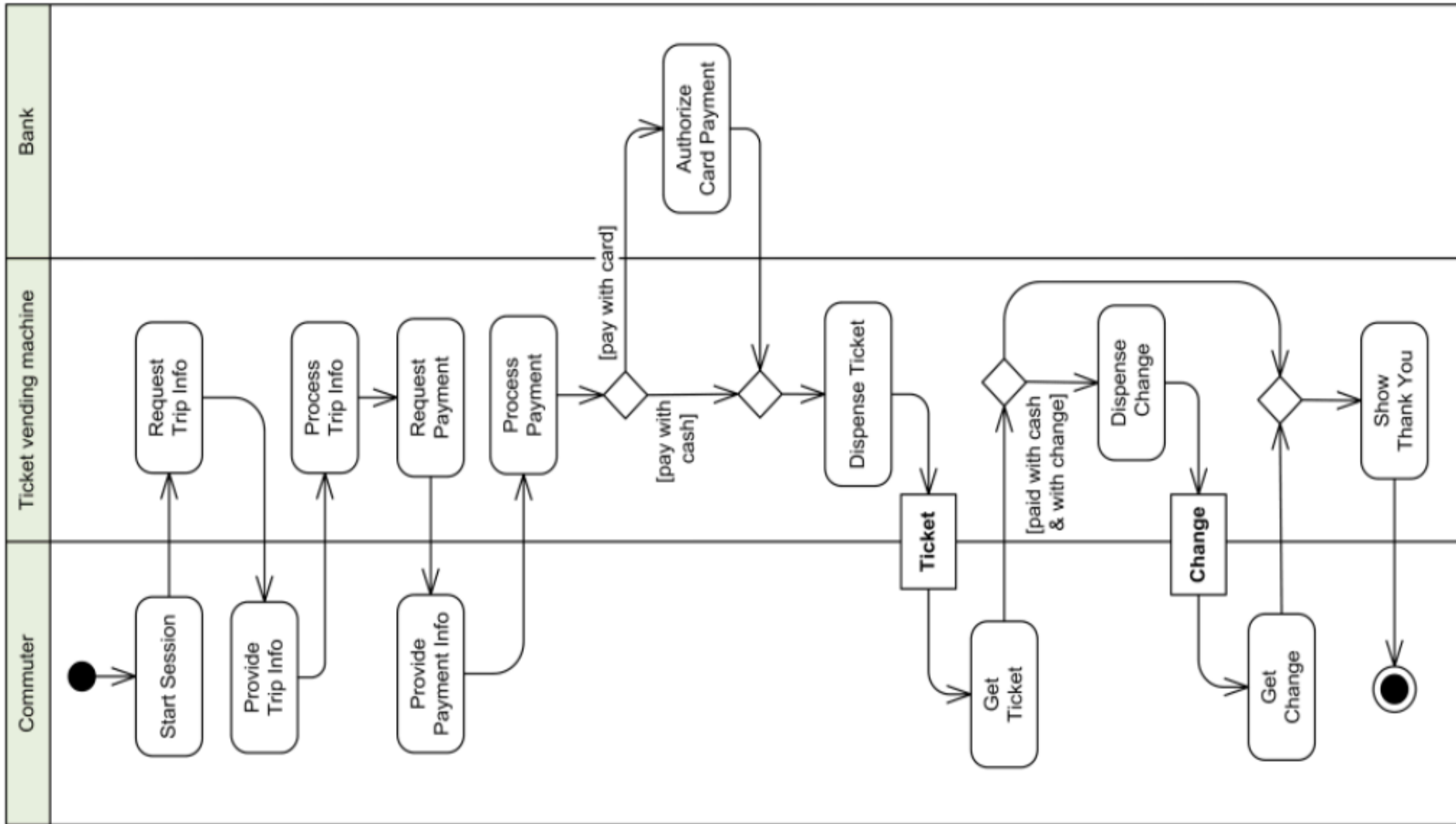
Example Context Diagram



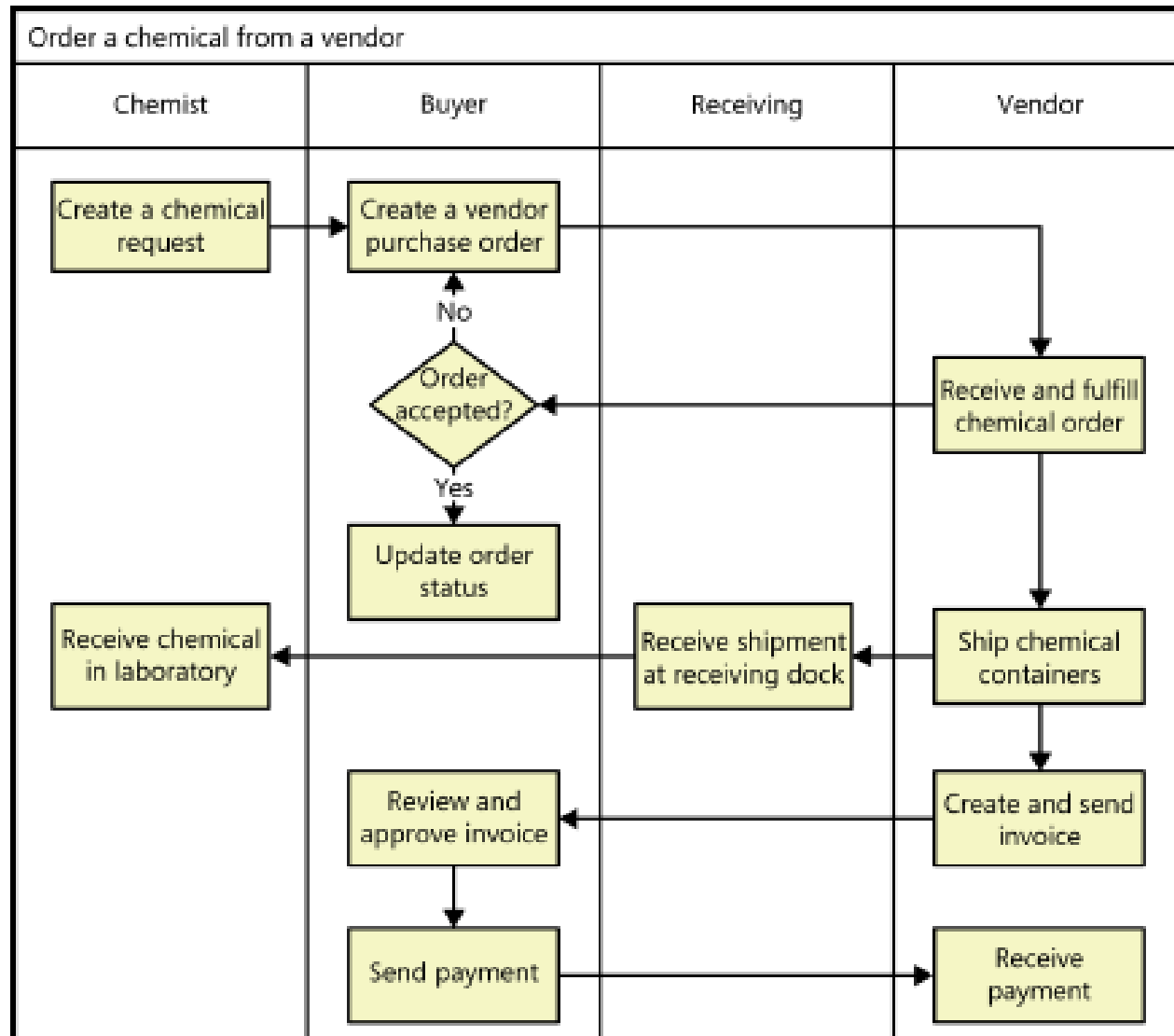
Example Level zero diagram






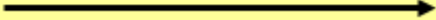
Swimlane Diagrams

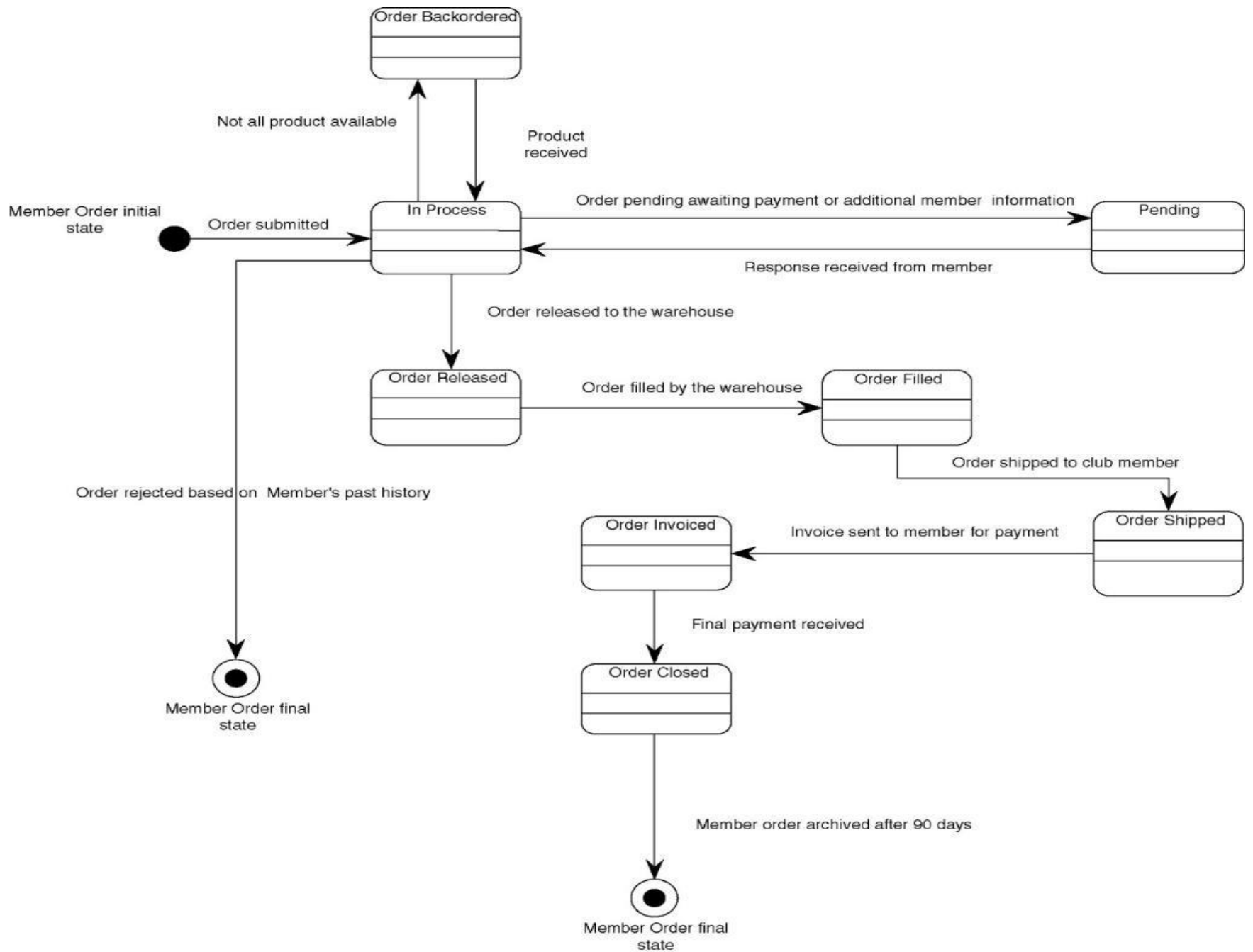


Swimlane Example



State Transition Diagram

A STATE	
AN INITIAL STATE	
A FINAL STATE	
AN EVENT	anEvent
A TRANSITION	



State Tables

- They show all the possible transitions between states in the form of matrix.

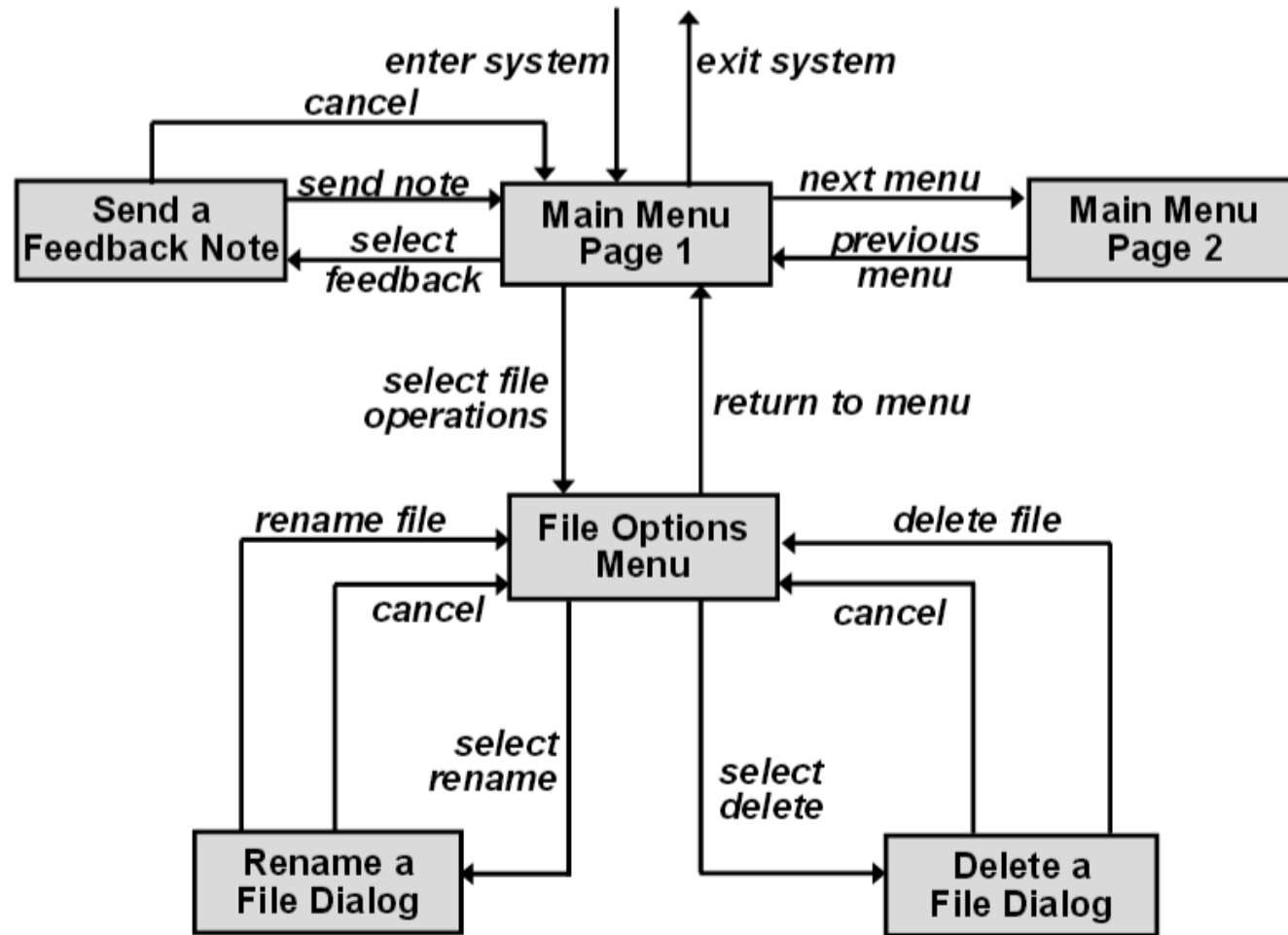
	In preparation	Postponed	Accepted	Placed	Back-Ordered	Fulfilled	Canceled
In Preparation	no	user saves incomplete request	system accepts valid request	no	no	no	no
Postponed	user retrieves incomplete request	no	no	no	no	no	no
Accepted	no	no	no	buyer places order with vendor	no	chemical stockroom fills request	requester cancels request
Placed	no	no	no	no	vendor places chemical on back-order	chemical received from vendor	buyer cancels vendor order
Back-Ordered	no	no	no	no	no	chemical received from vendor	buyer cancels vendor order
Fulfilled	no	no	no	no	no	no	no
Canceled	no	no	no	no	no	no	no

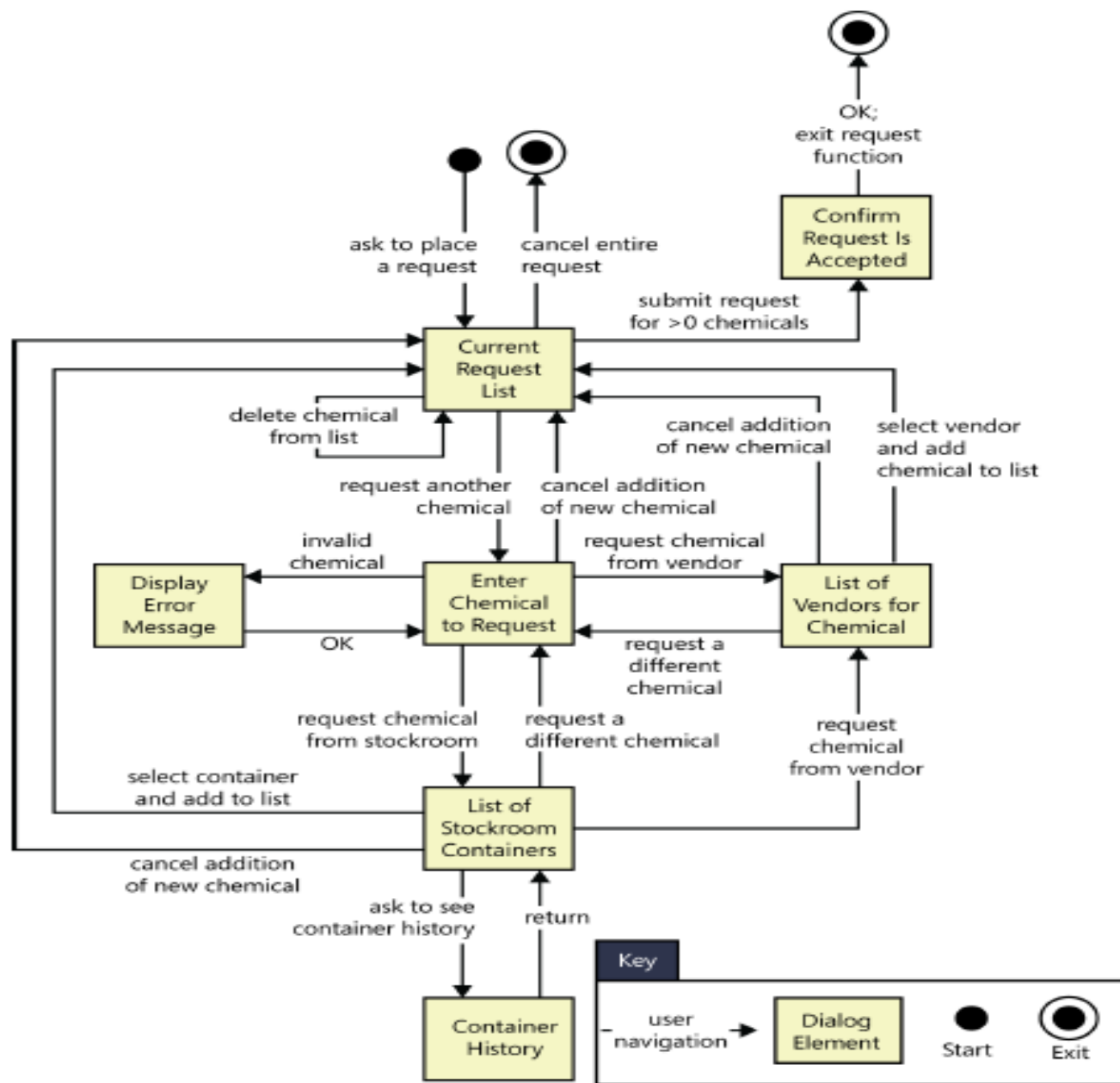
State table for a chemical request in the Chemical Tracking System.

Dialog Map

- The dialog map represents a user interface design at a high level of abstraction.
- Dialog element is represented as a state – Rectangle
- Each allowed navigation option – Arrow
- The condition that triggers user interface navigation is shown as text label on transition arrow.
- Several types of triggers:
 - User action
 - Data value
 - System condition
 - Some combination of these

User Interface Modeling Using a Dialog Map





A partial dialog map for the "Request a Chemical" use case from the Chemical Tracking System.

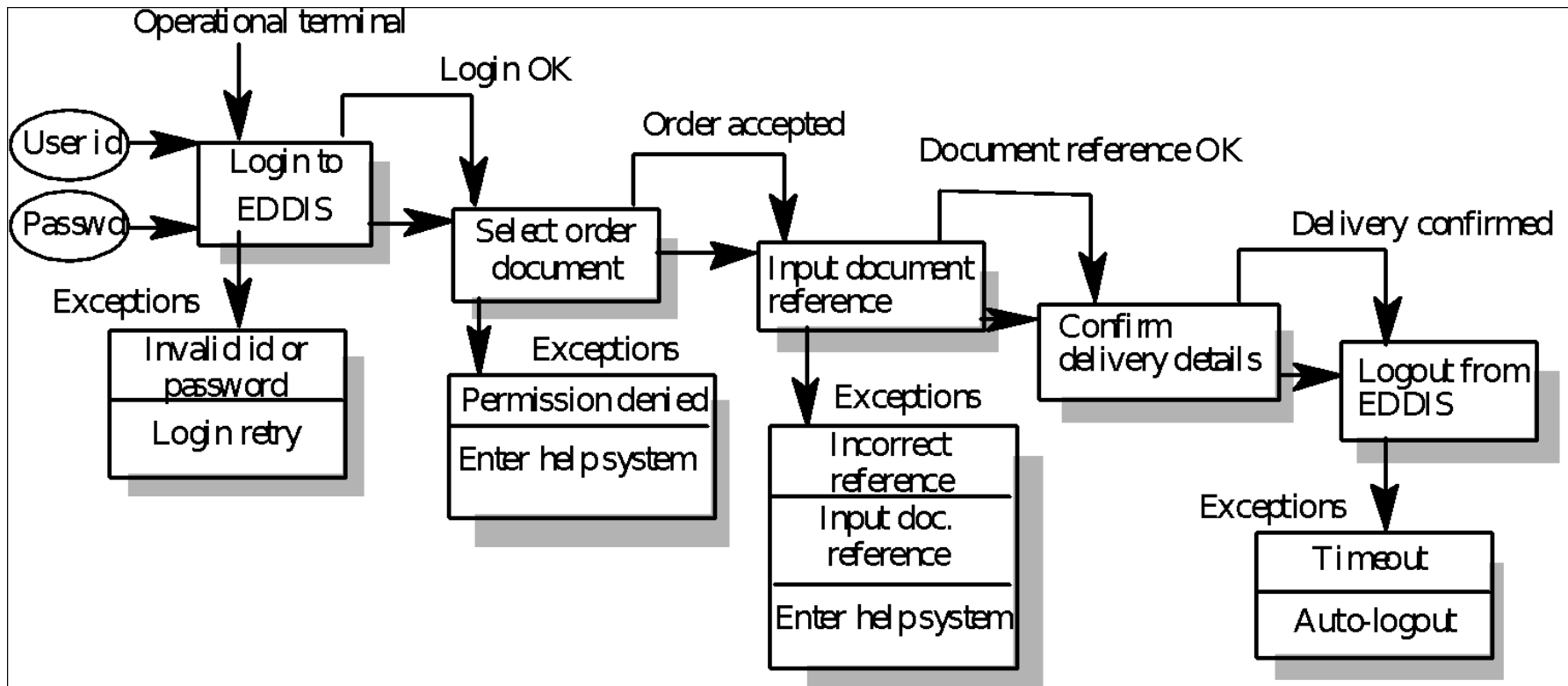
Scenarios

- Scenarios are stories which explain how a system might be used. They should include
 - a description of the system state before entering the scenario
 - the normal flow of events in the scenario
 - exceptions to the normal flow of events
 - information about concurrent activities
 - a description of the system state at the end of the scenario
- Scenarios are examples of interaction sessions which describe how a user interacts with a system
- Discovering scenarios exposes possible system interactions and reveals system facilities which may be required

Library scenario - document ordering

- Log on to EDDIS system
- Issue order document command
- Enter reference number of the required document
- Select a delivery option
- Log out from EDDIS
- This sequence of events can be illustrated in a diagram

Library Scenario



Decision Tables & Decision Trees

- A software system is often governed by complex logic, with various combinations of conditions leading to different system behaviors.

Example- Decision Table

- **Example: An insurance s/w calculates premium based on the following rules:**
 - If the insured is male, then premium is \$1500 per year
 - If the insured is female, then premium is \$1000 per year
 - If the insured's car has alarm, premium is reduced by 10%

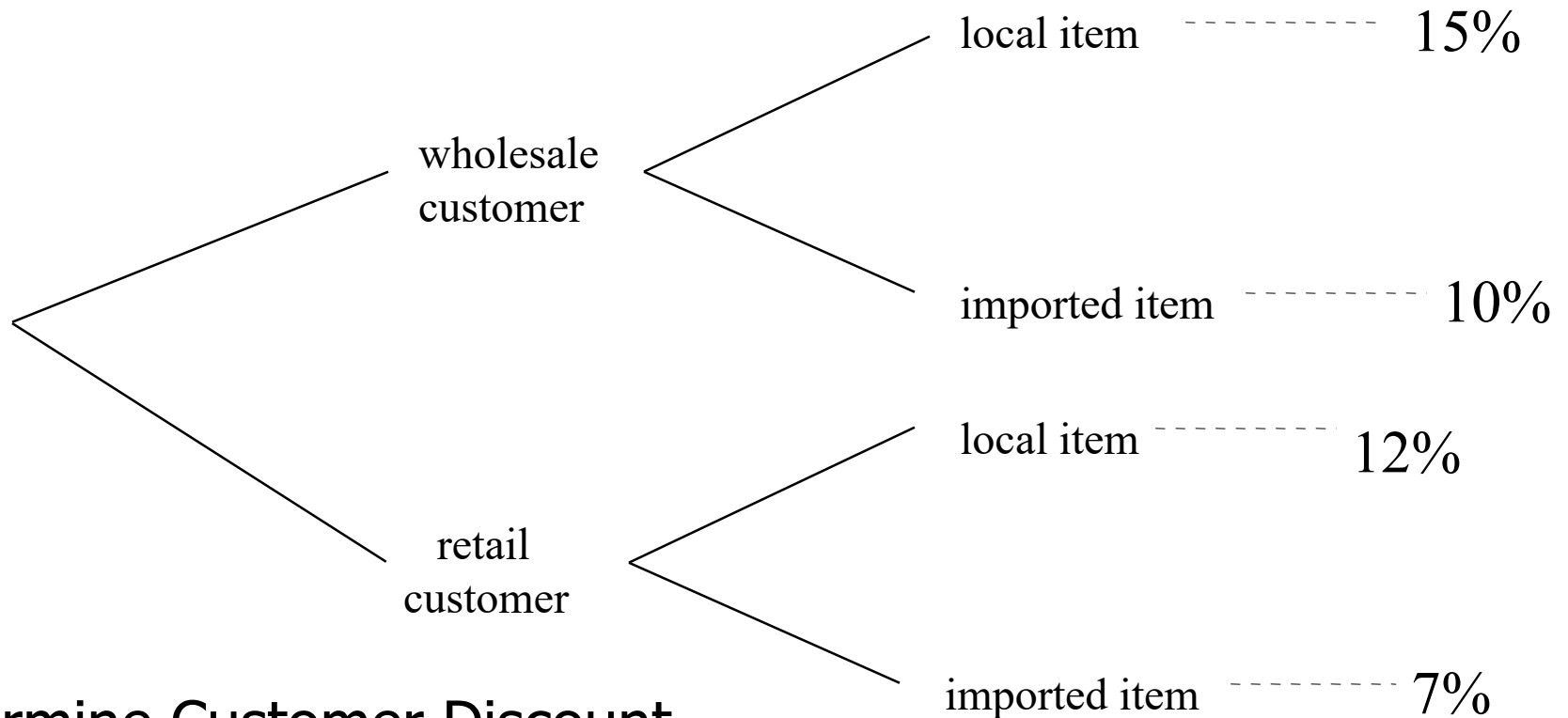
Condition/Rule	Rule 1	Rule 2	Rule 3	Rule 4	Rule 5	Rule 6	Rule 7	Rule 8
Male	T	T	T	T	F	F	F	F
Female	T	T	F	F	T	T	F	F
Car has alarm	T	F	T	F	F	T	T	F
Result	X	X	\$1350	\$1500	\$1000	\$900	X	X

Condition Stubs

Action Stubs

Rules

Decision Trees

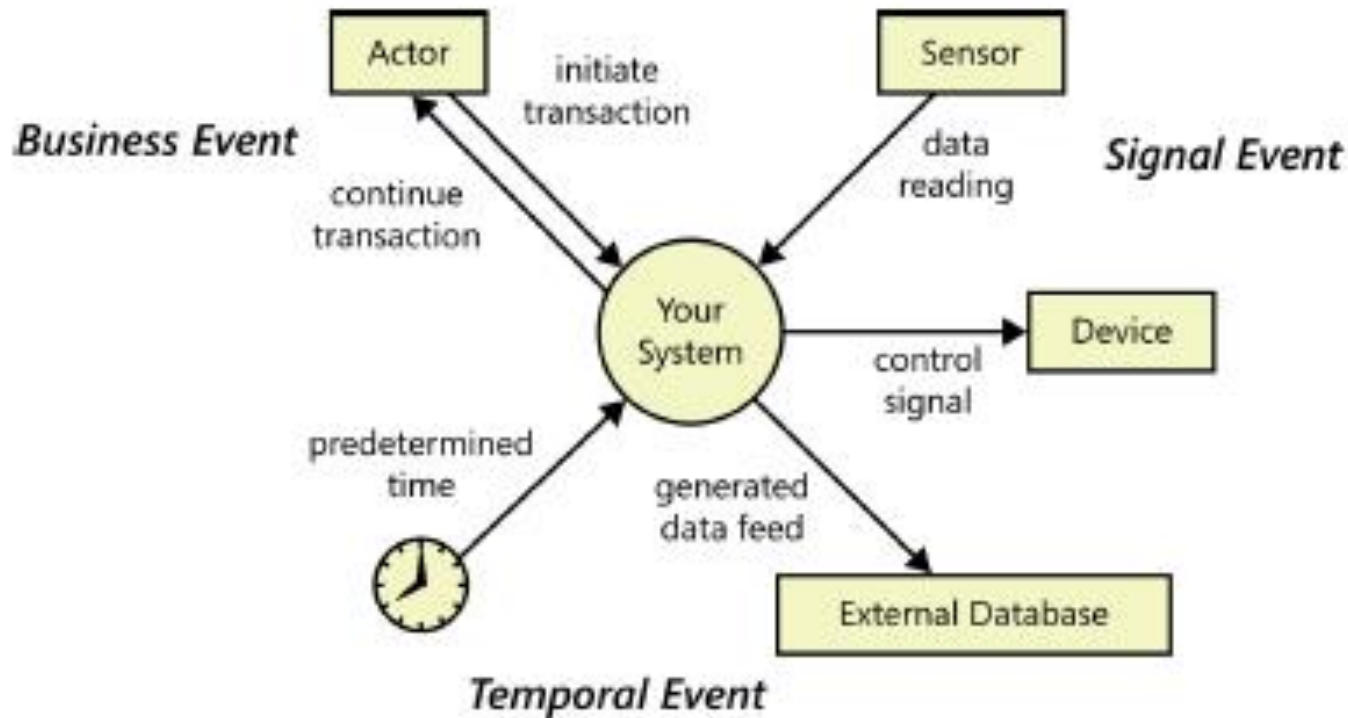


Determine Customer Discount

Event Response Table

- An event is some change or activity that takes place in a user's environment that stimulates a response from the software system.
 - **Business Event**
 - **Signal event**
 - **Temporal event**
- An **event response table** also called an event table or an event list lists all such events and the behavior of the system that is expected to exhibit in reaction to each event.

System Responses to Events



Systems respond to business, signal, and temporal events.

Event Response Table Example

Partial event-response table for an automobile windshield-wiper system

ID	Event	System state	System response
1	Set wiper control to low speed	Wiper off, on high speed, or on intermittent	Set wiper motor to low speed
2	Set wiper control to high speed	Wiper off, on low speed, or on intermittent	Set wiper motor to high speed
3	Set wiper control to off	Wiper on high speed, low speed, or intermittent	1. Complete current wipe cycle 2. Turn wiper motor off
4	Set wiper control to intermittent	Wiper off	1. Perform one wipe cycle 2. Read wipe time interval setting 3. Initialize wipe timer
5	Set wiper control to intermittent	Wiper on low speed or on high speed	1. Complete current wipe cycle 2. Read wipe time interval setting 3. Initialize wipe timer
6	Wipe time interval has passed since completing last cycle	Wiper on intermittent	Perform one wipe cycle at low speed setting
7	Change intermittent wiper interval	Wiper on intermittent	1. Read wipe time interval setting 2. Initialize wipe timer
8	Change intermittent wiper interval	Wiper off, on high speed, or on low speed	No response
9	Immediate wipe signal received	Wiper off	Perform one low-speed wipe cycle

Specifying Data Requirements

Data Modeling

Data modeling – a technique for organizing and documenting a system's data. Sometimes called *database modeling*.

Entity relationship diagram (ERD) – a data model utilizing several notations to depict data in terms of the entities and relationships described by that data.

Data Modeling Concepts: Entity

- Entity – a class of persons, places, objects, events or concepts about which we need to capture and store data.



Attributes

Attribute – a descriptive property or characteristic of an entity. Synonyms include *element*, *property*, and *field*.

Compound attribute – an attribute that consists of other attributes. Synonyms in different data modeling languages are numerous: concatenated attribute, composite attribute, and data structure.

STUDENT

Name

.Last Name

.First Name

.Middle Initial

Address

.Street Address

.City

.State or Province

.Country

.Postal Code

Phone Number

.Area Code

.Exchange Number

.Number Within Exchange

Date of Birth

Gender

Race

Major

Grade Point Average

Identification

Key – an attribute, or a group of attributes, that assumes a unique value for each entity instance. It is sometimes called an *identifier*.

Concatenated key

Candidate key

Primary key

Alternate key.

STUDENT

Student Number
(Primary Key)

Social Security Number
(Alternate Key)

Name

.Last Name

.First Name

.Middle Initial

Address

.Street Address

.City

.State or Province

.Country

.Postal Code

Phone Number

.Area Code

.Exchange Number

.Number Within Exchange

Date of Birth

Gender (Subsetting Criteria 1)

Race (Subsetting Criteria 2)

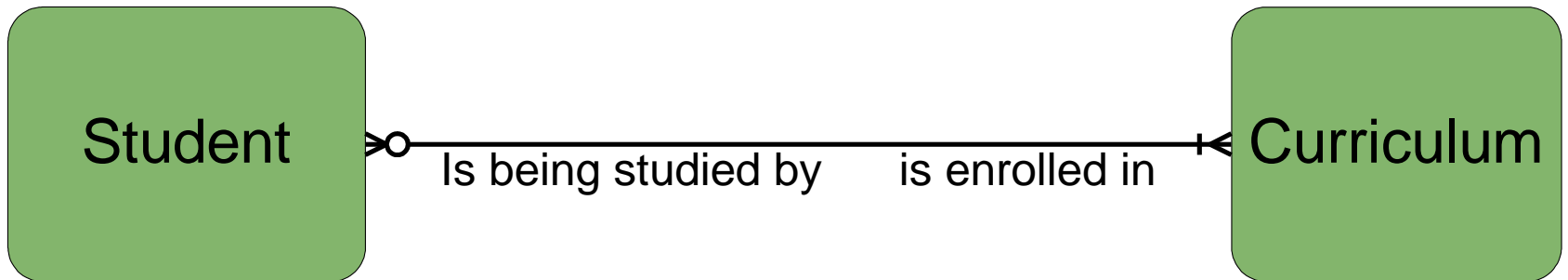
Major (Subsetting Criteria 3)

Grade Point Average

Relationships

Relationship – a natural business association that exists between one or more entities.

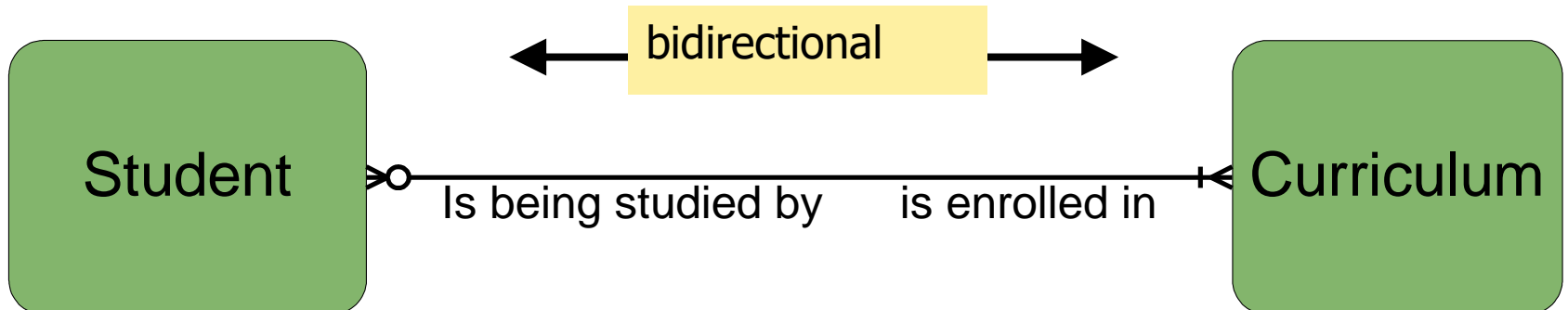
The relationship may represent an event that links the entities or merely a logical affinity that exists between the entities.



Cardinality

Cardinality – the minimum and maximum number of occurrences of one entity that may be related to a single occurrence of the other entity.

Because all relationships are bidirectional, cardinality must be defined in both directions for every relationship.



Degree

Degree – the number of entities that participate in the relationship.

A relationship between two entities is called a *binary relationship*.

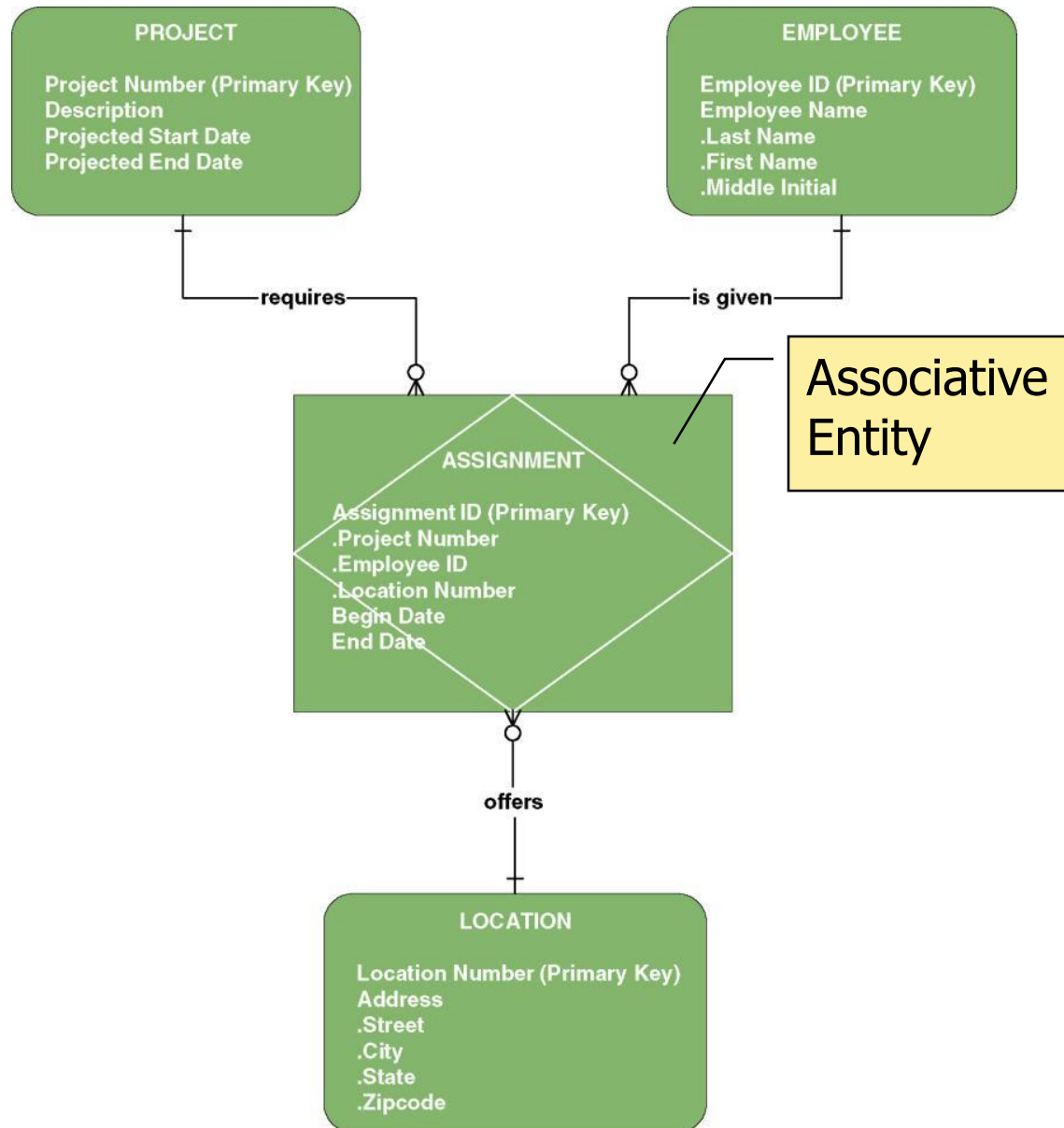
A relationship between different instances of the same entity is called a *recursive relationship*.

A relationship between three entities is called a *3-ary* or *ternary relationship*.

Data Modeling Concepts: Degree

Associative entity – an entity that inherits its primary key from more than one other entity (called parents).

Each part of that concatenated key points to one and only one instance of each of the connecting entities.

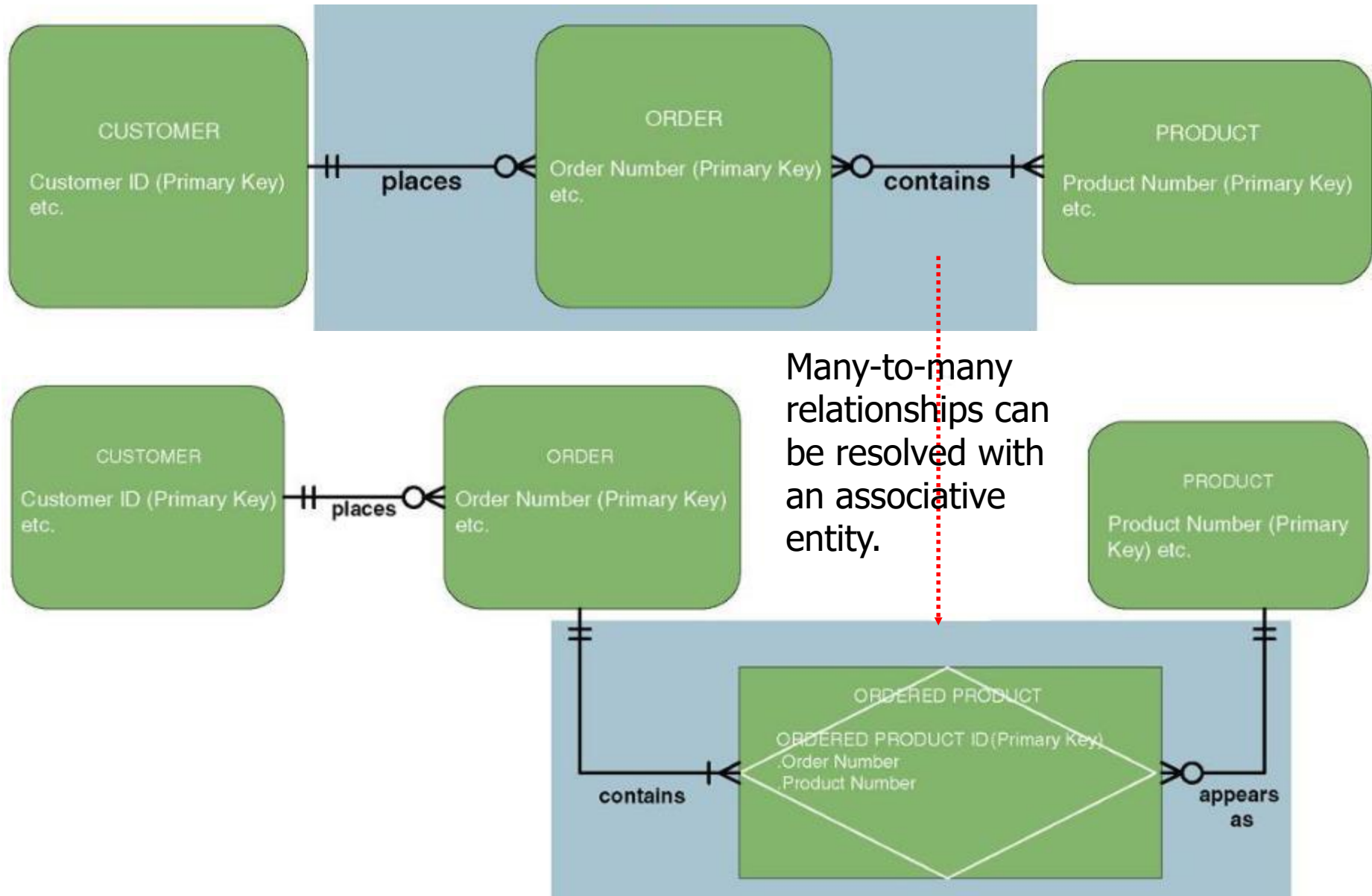


Foreign Keys

Foreign key – a primary key of an entity that is used in another entity to identify instances of a relationship.

- A foreign key is a primary key of one entity that is contributed to (duplicated in) another entity to identify instances of a relationship.
- A foreign key always matches the primary key in the another entity
- A foreign key may or may not be unique (generally not)
- The entity with the foreign key is called the child.
- The entity with the matching primary key is called the parent.

Resolving Nonspecific Relationships



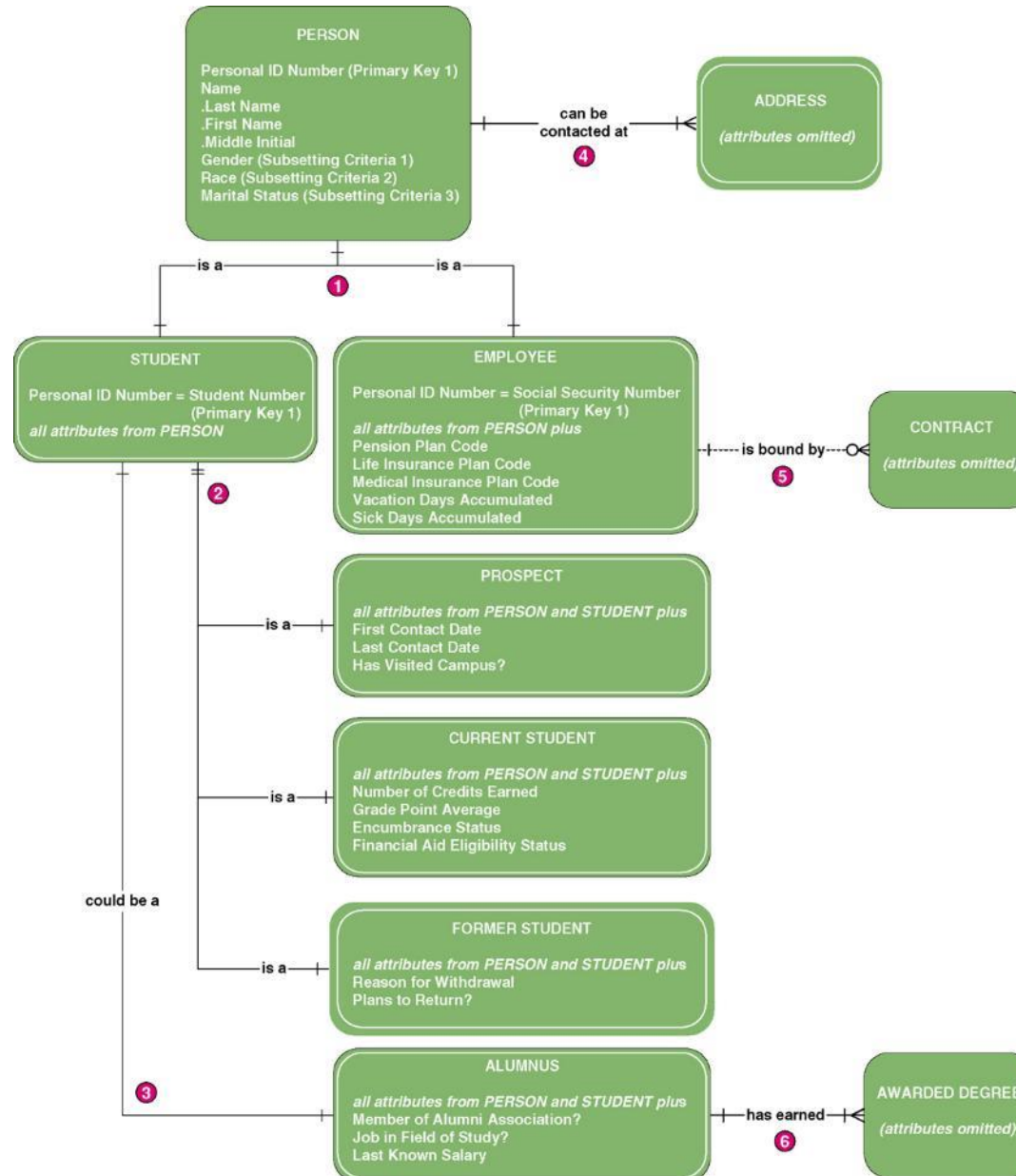
Generalization

Generalization – a concept wherein the attributes that are common to several types of an entity are grouped into their own entity.

Supertype – an entity whose instances store attributes that are common to one or more entity subtypes.

Subtype – an entity whose instances may inherit common attributes from its entity supertype

Generalization Hierarchy



Data Analysis & Normalization

Data analysis – a technique used to improve a data model for implementation as a database.

Goal is a simple, nonredundant, flexible, and adaptable database.

Normalization – a data analysis technique that organizes data into groups to form nonredundant, stable, flexible, and adaptive entities.

Normalization: 1NF, 2NF, 3NF

First normal form (1NF) – an entity whose attributes have no more than one value for a single instance of that entity

- Any attributes that can have multiple values actually describe a separate entity, possibly an entity and relationship.

Second normal form (2NF) – an entity whose nonprimary-key attributes are dependent on the full primary key.

- Any nonkey attributes that are dependent on only part of the primary key should be moved to any entity where that partial key is actually the full key. This may require creating a new entity and relationship on the model.

Third normal form (3NF) – an entity whose nonprimary-key attributes are not dependent on any other non-primary key attributes.

- Any nonkey attributes that are dependent on other nonkey attributes must be moved or deleted. Again, new entities and relationships may have to be added to the data model.

Data Dictionary

- A **data dictionary** is a collection of detailed information about the data entities used in an application.

Data Element	Description	Composition or Data Type	Length	Values
Chemical Request	request for a new chemical from either the Chemical Stockroom or a vendor	Request ID + Requester + Request Date + Charge Number + 1:10(Requested Chemical)		
Delivery Location	the place to which requested chemicals are to be delivered	Building + Lab Number + Lab Partition		
Number of Containers	number of containers of a given chemical and size being requested	Positive integer	3	
Quantity	amount of chemical in the requested container	numeric	6	
Quantity Units	units associated with the quantity of chemical requested	alphabetic characters	10	grams, kilograms, milligrams, each
Request ID	unique identifier for a request	integer	8	system-generated sequential integer, beginning with 1

Data Dictionary - Entities

- A set of information describing the contents, format, and structure of a database and the relationship between its elements, used to control access to and manipulation of the database.
- Example:

Field Name	Data Type	Size	Bytes	Constraint	Range		Alias	E.g. of legitimate value

Data Structure Notation /data Dictionary for Data Flow

1. = consist of or composed of
2. + and or sequence
3. [] selection,
4. {} Repetition,
5. () optional

A data structure for a data flow

```
ORDER=
  ORDER NUMBER +
  ORDER DATE+
  [ PERSONAL CUSTOMER NUMBER,
    CORPORATE ACCOUNT NUMBER]+
  SHIPPING ADDRESS=ADDRESS+
  (BILLING ADDRESS=ADDRESS)+
  1 {PRODUCT NUMBER+
    PRODUCT DESCRIPTION+
    QUANTITY ORDERED+
    PRODUCT PRICE+
    PRODUCT PRICE SOURCE+
    EXTENDED PRICE } N+
  SUM OF EXTENDED PRICES+
  PREPAID AMOUNT+
  (CREDIT CARD NUMBER+EXPIRATION
  DATE)
```

Data Analysis – CRUD Matrix

- It is a rigorous data analysis technique for detecting missing requirements.
- CRUD – Create, Read, Update & Delete
- A CRUD matrix correlates system actions with data entities to show where and how each significant data entity is created, read, updated and deleted.

Use Case \ Entity	Order	Chemical	Requester	Vendor Catalog
Place Order	C	R	R	R
Change Order	U, D		R	R
Manage Chemical Inventory		C, U, D		
Report on Orders	R	R	R	
Edit Requesters			C, U	

Sample CRUD matrix for the Chemical Tracking System.

Specifying reports

- Many applications generate reports from one or more databases, files, or other information sources.
- Reports can consist of traditional tabular presentations of rows and columns of data, charts and graphs of all types, or any combination.

Report Specification Template

Report Element	Element Description
Report ID	Number, code, or label used to identify or classify the report
Report Title	<ul style="list-style-type: none">■ Name of the report■ Positioning of the title on the page■ Include query parameters used to generate the report (such as date range)?
Report Purpose	Brief description of the project, background, context, or business need that led to this report
Decisions Made from Report	The business decisions that are made using information in the report
Priority	The relative priority of implementing this reporting capability
Report Users	User classes who will generate the report or use it to make decisions
Data Sources	The applications, files, databases, or data warehouses from which data will be extracted
Frequency and Disposition	<ul style="list-style-type: none">■ Is the report static or dynamic?■ How frequently is the report generated: weekly, monthly, on demand?■ How much data is accessed, or how many transactions are included, when the report is generated?■ What conditions or events trigger generation of the report?■ Will the report be generated automatically? Is manual intervention required?■ Who will receive the report? How is it made available to them (displayed in an application, sent in email, printed, viewed on a mobile device)?

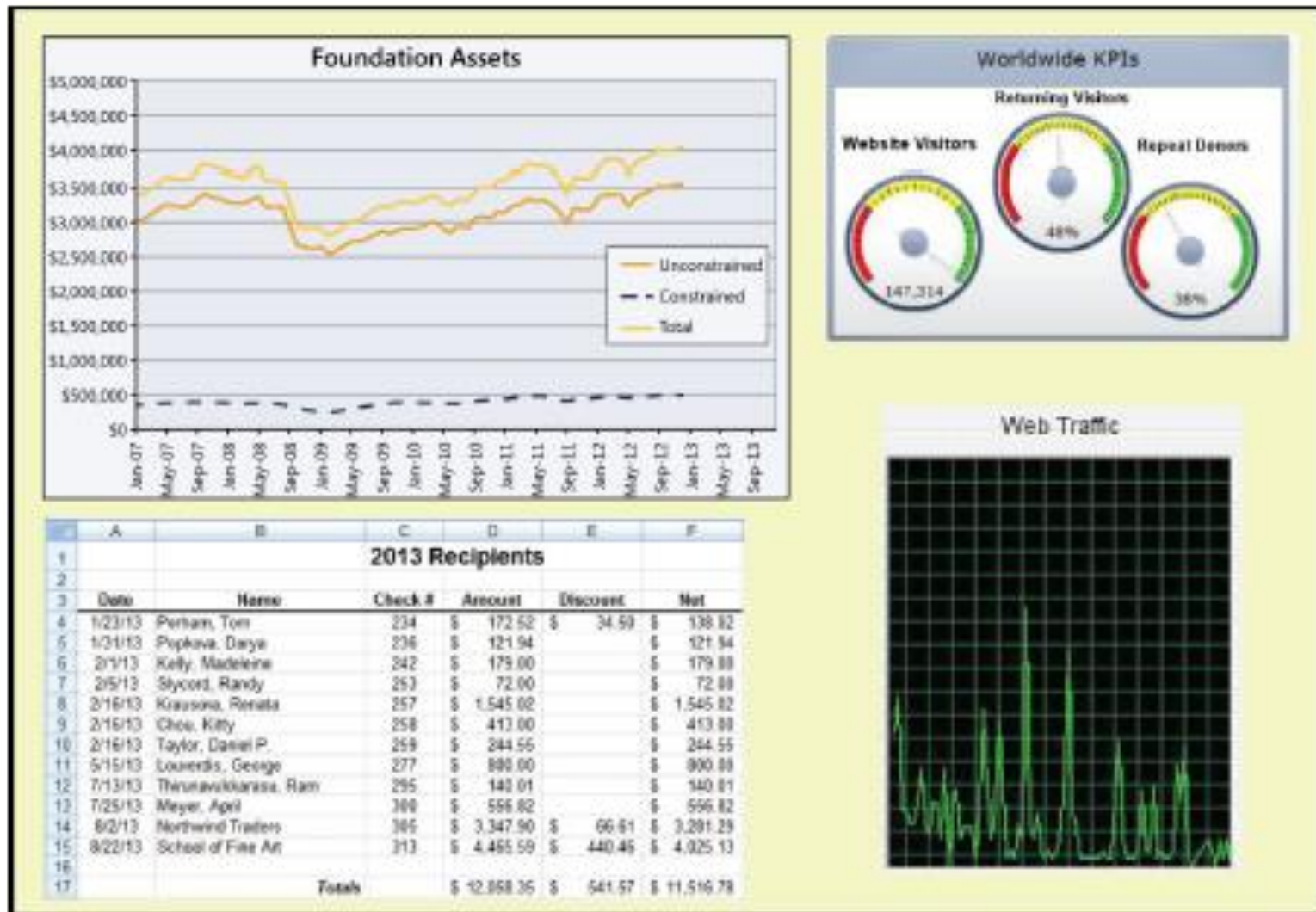
Report Specification Template

Report Element	Element Description
Latency	<ul style="list-style-type: none"> How quickly must the report be delivered to users when requested? How current must the data be when the report is run?
Visual Layout	<ul style="list-style-type: none"> Landscape or portrait Paper size (or type of printer) to be used for hard-copy reports If the report includes graphs, define the type(s) of each graph, its appearance, and parameters: titles, axis scaling and labels, data sources, and so on
Header and Footer	<p>The following items are among those that could be positioned somewhere in the report header or footer. For each element included, specify the location on the page and its appearance, including font face, point size, text highlighting, color, case, and text justification. When a title or other content exceeds its allocated space, should it be truncated, word-wrapped to the next line, or what?</p> <ul style="list-style-type: none"> Report title Page numbering and format (such as "Page x" or "Page x of y") Report notes (such as "The report excludes employees who worked for the company for less than one month.") Report run timestamp Name of the person who generated the report Data source(s), particularly in a data warehousing application that consolidates data from multiple sources Report begin and end dates Organization identification (company name, department, logo, other graphics) Confidentiality statement or copyright notice
Report Body	<ul style="list-style-type: none"> Record selection criteria (logic for what data to select and what to exclude) Fields to include User-specified text or parameters to customize field labels Column and row heading names and formats: text, font, size, color, highlighting, case, justification Column and row layout of data fields, or graph positioning and parameters for charts or graphs Display format for each field: font, size, color, highlighting, case, justification, alignment, numeric rounding, digits and formatting, special characters (\$, %, commas, decimals, leading or trailing pad characters) How numeric and text field overflows should be handled Calculations or other transformations that are performed to generate the data displayed Sort criteria for each field Filter criteria or parameters used to restrict the report query prior to running the report Grouping and subtotals, including formatting of totals or subtotal breakout rows Paging criteria
End-of-Report Indicator	Appearance and position of any indicator that appears at the end of the report
Interactivity	<ul style="list-style-type: none"> If the report is dynamic or is generated interactively, what options should the user have to modify the contents or appearance of the initially generated report (expand and collapse views, link to other reports, drill down to data sources)? What is the expected persistence of report settings between usage sessions?
Security Access Restrictions	Any limitations regarding which individuals, groups, or organizations are permitted to generate or view the report or which data they are permitted to select for inclusion

Dashboard Reporting

- A *dashboard* is a screen display or printed report that uses multiple textual and/or graphical representations of data to provide a consolidated, multidimensional view of what is going on in an organization or a process.
- Companies often use dashboards to pull together information about sales, expenses, key performance indicators (KPIs), and the like.
- Certain displays in a dashboard might be dynamically updated in real time as input data changes.

Dashboard Reporting



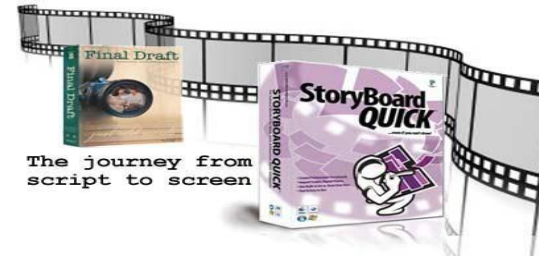
Hypothetical reporting dashboard for a charitable foundation.

Eliciting & Analysis for Dashboard

- Determine what information the dashboard users need for making specific decisions or choices.
- Identify the sources of all the data to be presented so you can ensure that the application has access to those feeds and you know whether they are static or dynamic
- Choose the most appropriate type of display for each set of related data.
- Determine optimal layout and relative sizing of the various displays in the dashboard based on how the user will absorb and apply the information.
- Specify the details of each display in the dashboard.
- **Prototyping a dashboard** is an excellent way to work with stakeholders to meet user needs.

Reference Guiding Topics for Analysis

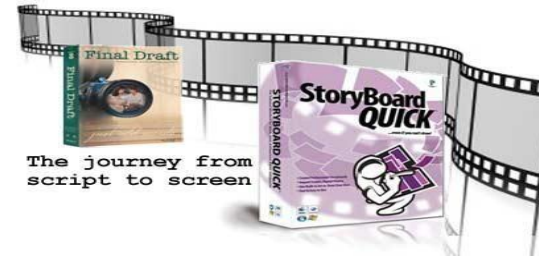
Storyboard



- ❑ A Storyboard is a logical and conceptual description of system functionality for a specific scenario, including the interaction required between the users and the system
- ❑ A Storyboard "tells a specific story"
- ❑ The goal in all techniques is to interact with the user
 - ✓ Helps to Bridge the gap between the user and the developer
 - ✓ Nonetheless, most of these interpretations agree that the purpose of storyboarding is to gain an early reaction from the users on the concepts proposed for the application.
 - ✓ With storyboarding, the user's reaction can be observed very early in the lifecycle. In so doing, storyboards offer an effective technique for addressing the "Yes, But" syndrome.
 - ✓ Storyboarding is also very effective for "Blank page" Syndrome



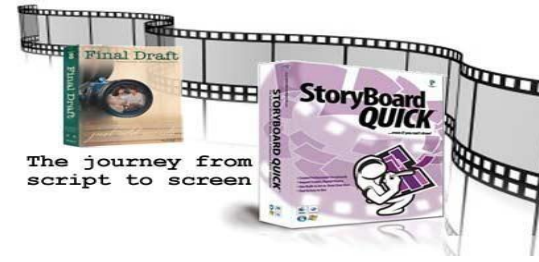
Benefits of Storyboard



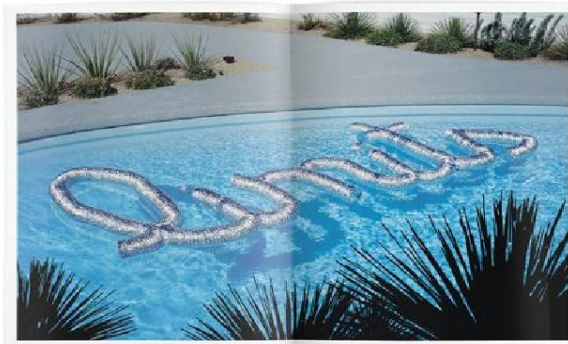
- ❑ A storyboard serves multiple purposes
 - ✓ Provides an early review of the User Interface of the system
 - ✓ Is extremely inexpensive
 - ✓ Is user friendly, informal, and interactive
 - ✓ Is easy to create and easy to modify



Limitations of Storyboard

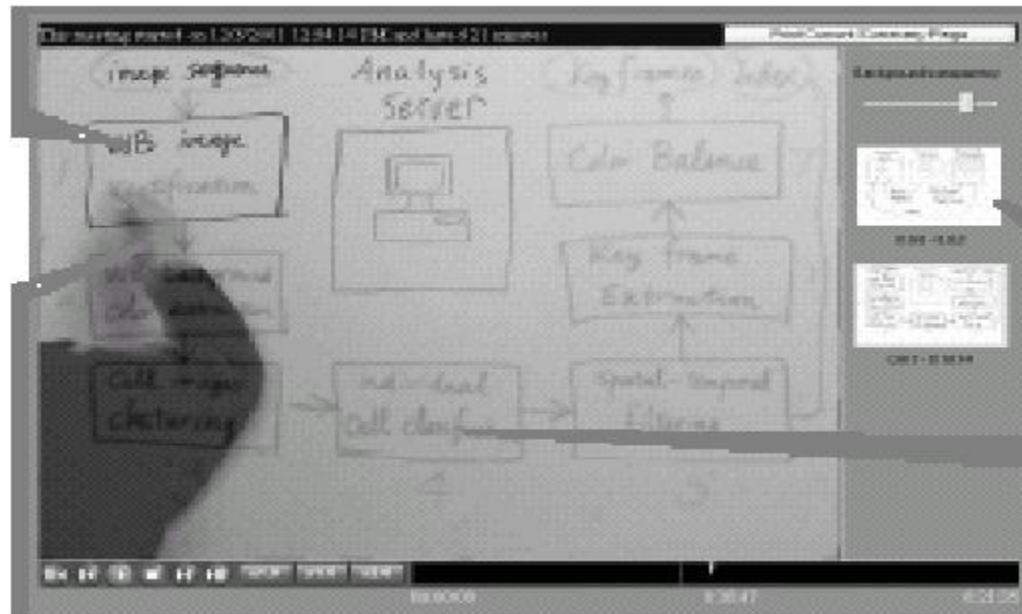


- One of the biggest problems with storyboards is that they can become obsolete very quickly. User interfaces originally defined often change over time, and that creates a maintenance burden.



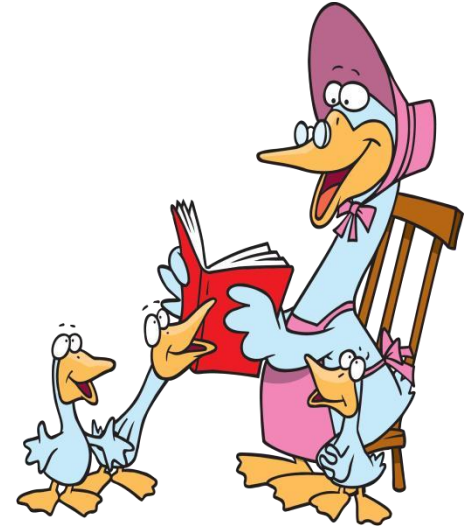
Types of Storyboards

- ❑ Passive Storyboards
- ❑ Active Storyboards
- ❑ Interactive Storyboards



Passive Storyboards

- ❑ Passive storyboards tell a story to the user.
- ❑ They can consist of sketches, pictures, screen presentations, shots, or sample application outputs.
- ❑ In a passive storyboard, the analyst plays the role of the system with a "When the role do this, this happens" explanation, and simply walks the user through the storyboard,



Active Storyboards

- ❑ Active storyboards try to make the user see "a movie that hasn't actually been produced yet."
- ❑ Active storyboards are animated or automated, perhaps by an automatically sequencing slide presentation, an animation tool, a recorded computer script or simulation, or even a homemade movie
- ❑ Active storyboards provide an automated description of the way the system behaves in a typical usage or operational scenario.



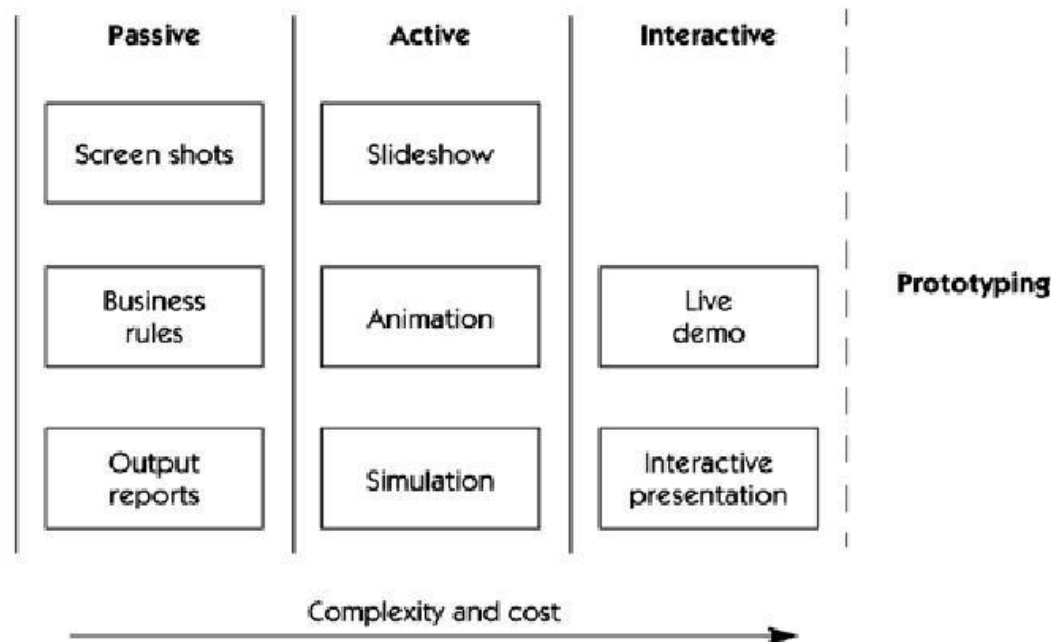
Interactive Storyboards

- ❑ Interactive storyboards let the user experience the system in as realistic a manner as practical.
- ❑ They require participation by the user.
- ❑ Interactive storyboards can be simulations or mock-ups or can be advanced to the point of throwaway code.
- ❑ An advanced, interactive storyboard built out of throwaway code can be very close to a throwaway prototype.



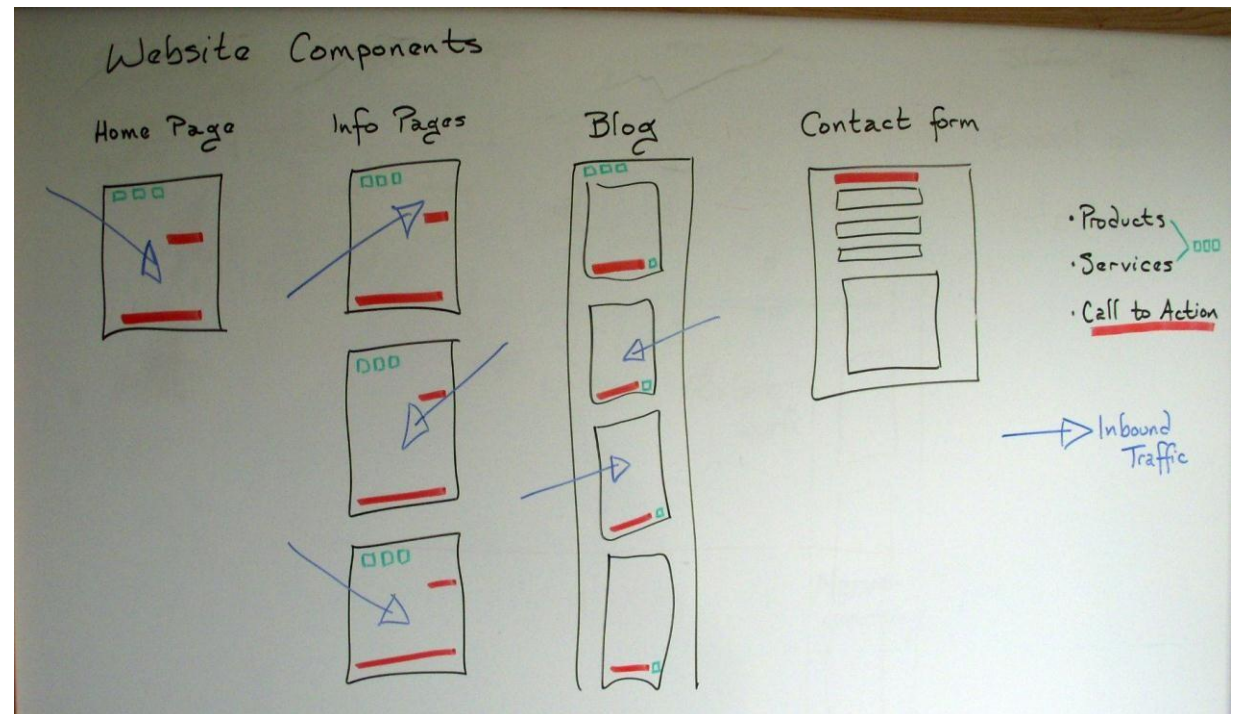
Types of Storyboards

- As the below figure shows, these three storyboarding techniques offer a range of possibilities ranging from sample outputs to live interactive demos.
- Indeed, the boundary between advanced storyboards and early product prototypes is a fuzzy one.



Tools for Storyboarding

- ❑ The tools and techniques for storyboarding can be as varied as the team members' and users' imaginations will allow.
- ❑ Passive-storyboarding constructs have been made out of tools as simple as paper and pencil or Post-it notes.



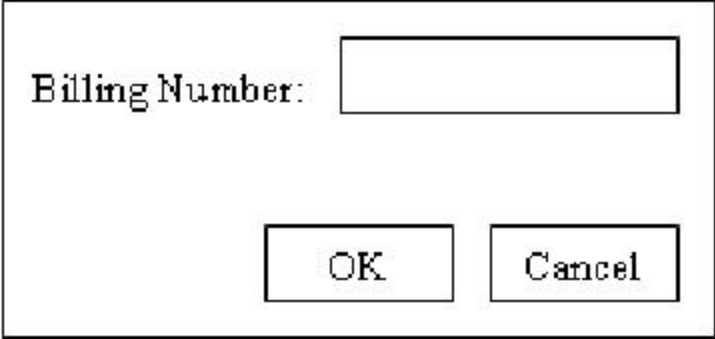
Who uses Storyboards?

The following people use the Storyboards:

- ❑ **System analysts**, to explore, clarify, and capture the behavioral interaction envisioned by the user as part of requirements elicitation.
- ❑ **user-interface designers**, to design the user interface and to build a prototype of the user interface;
- ❑ **designers** of the classes that provide the user interface functionality. They use this information to understand the system's interactions with the user, so they can properly design the classes.
- ❑ **those who test** to test the system's features.
- ❑ **the manager** to plan and follow up the analysis & design work.



Examples



A simple dialog box with a rectangular border. Inside, the text "Billing Number:" is positioned to the left of a rectangular text input field. Below the input field, there are two rectangular buttons: "OK" on the left and "Cancel" on the right.

- Imagine that in reviewing this visual storyboard, our end-user accountant says, "Well, actually, billing numbers are divided into two parts, the year and a unique number."
- This drawing shows only one field for the account number." Then he adds: "And by the way, I don't want to enter the year all the time, so please initialize this value with the current year, which I can overwrite if necessary."

END OF TOPIC # 8

-COMING UP!!!!!!

-Stakeholder Analysis

-Negotiation & Prioritizing

-Non Functional Requirements

-Risk Reduction & Prototyping