SOFTWARE DESIGN & ARCHITECTURE

(Lecture-2)

USAMA MUSHARAF

LECTURER (Department of Computer Science)

FAST-NUCES PESHAWAR

CONTENT

- Agile Development
- Requirement Engineering
  - Analysis Modeling

AGILE DEVELOPMENT

WHAT IS "AGILE"?

Success Rate

$\llcorner$ follow

- Agile is a philosophy or a way of thinking guided by some **values and principles**.

WHAT IS "AGILITY"?

✓

- Effective (rapid and adaptive) response to change
- Effective communication among all stakeholders

Fast

Yielding …

- Rapid, incremental delivery of software

AGILE VALUES

✓ ✓

1. Individuals and Interactions over Processes and Tools

✓

2. Working Software over Comprehensive Documentation

✓ SRS

3. Customer Collaboration over Contract Negotiation
4. Responding to change over Following a Plan

Iteration
2-Weeks

Stakeholders
- Customer
- Programmer

→ Working Module

PRINCIPLES OF AGILE METHODS

1. Customer Involvement
2. Welcome Changes
3. Collaboration

AGILE PROCESS MODELS

- ✓ **Extreme Programming (XP)**
- ✓ **Scrum**
- Adaptive Software Development
- Dynamic System Development Method (DSDM)
- Crystal
- Feature Driven Development
- Agile Modeling (AM)

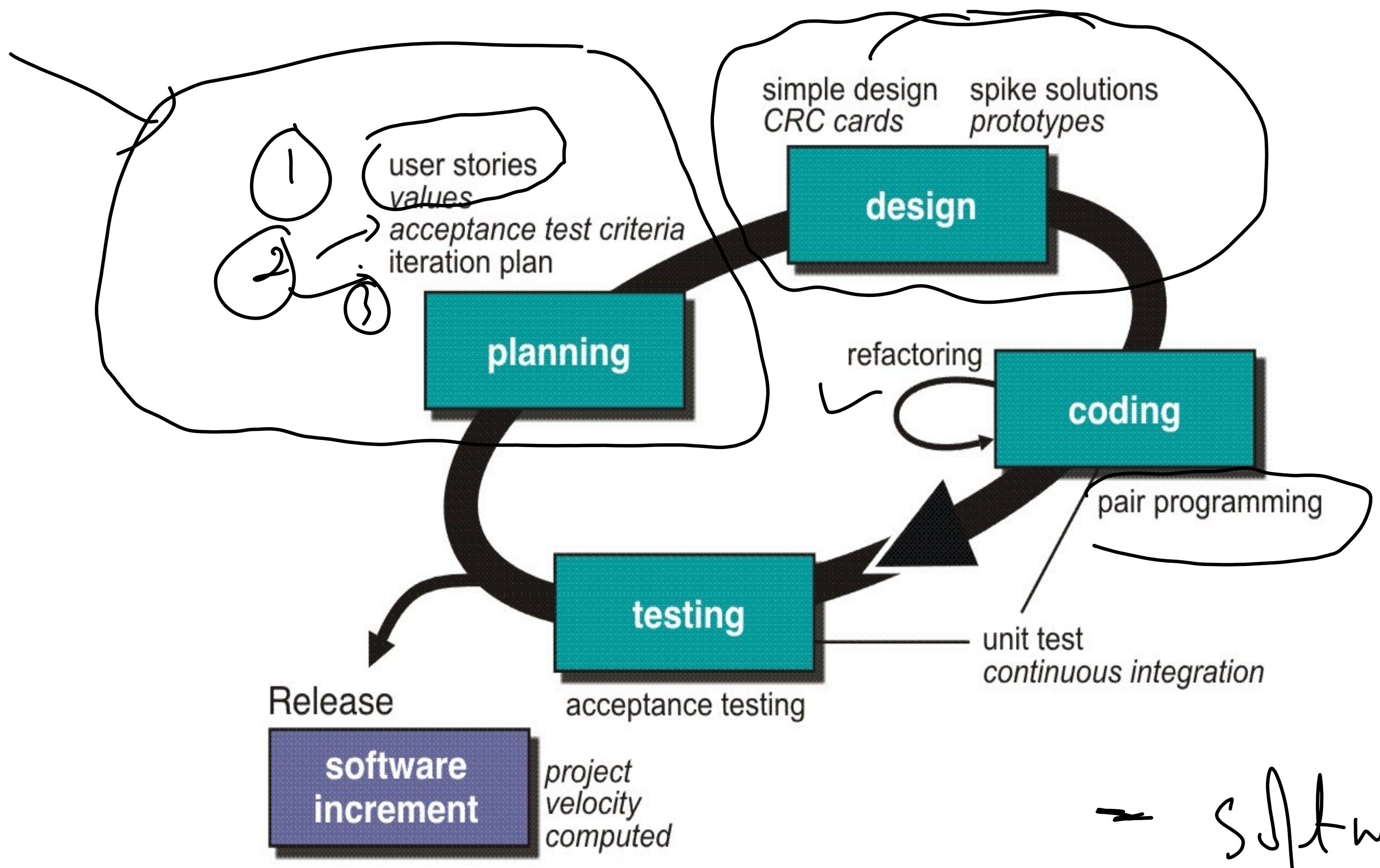
EXTREME PROGRAMMING (XP)

- Perhaps one of the well-known and most widely used agile method.
- Extreme Programming (XP) takes an 'extreme' approach to iterative development.
  - New versions may be built several times per day;
  - Increments are delivered to customers every 2 weeks;
  - All tests must be run for every build and the build is only accepted if tests run successfully.

Pair Team

$\boxed{\text{VCS}}$ — Git / Github

EXTREME PROGRAMMING (XP)



Recommending
×
Rules

Security

- Software Re-Engineering

EXTREME PROGRAMMING (XP)

XP Planning

- Begins with the creation of user stories
- Agile team assesses each story and assigns a cost
- Stories are grouped to for a deliverable increment
- A commitment is made on delivery date

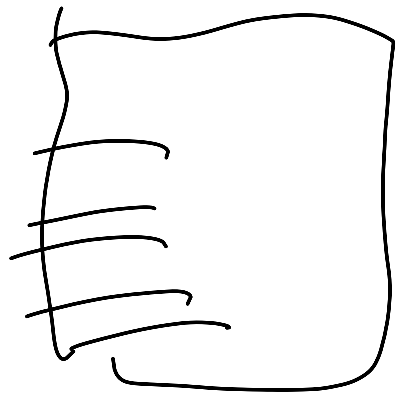
REQUIREMENTS SCENARIOS

- In XP, user requirements are expressed as scenarios or user stories.
- These are written on cards and the development team break them down into implementation tasks. These tasks are the basis of schedule and cost estimates.
- The customer chooses the stories for inclusion in the next release based on their priorities and the schedule estimates.

STORY CARD FOR DOCUMENT DOWNLOADING

Target Points

List



Payment

Downloading and printing an article

First, you select the article that you want from a displayed list. You then have to tell the system how you will pay for it - this can either be through a subscription, through a company account or by credit card.

After this, you get a copyright form from the system to fill in and, when you have submitted this, the article you want is downloaded onto your computer.

You then choose a printer and a copy of the article is printed. You tell the system if printing has been successful.

If the article is a print-only article, you canÕ t keep the PDF version so it is automatically deleted from your computer .

EXTREME PROGRAMMING (XP)

XP Design

- Follows the KIS (keep it simple) principle
- Encourage the use of CRC (class-responsibility-cards) cards
- For difficult design problems, suggests the creation of spike solutions — a design prototype
- Encourages refactoring — an iterative refinement of the internal program design

EXTREME PROGRAMMING (XP)

CRC Cards:

Class-responsibility-collaboration (CRC) cards are a tool used in the design of object-oriented software.

EXTREME PROGRAMMING (XP)

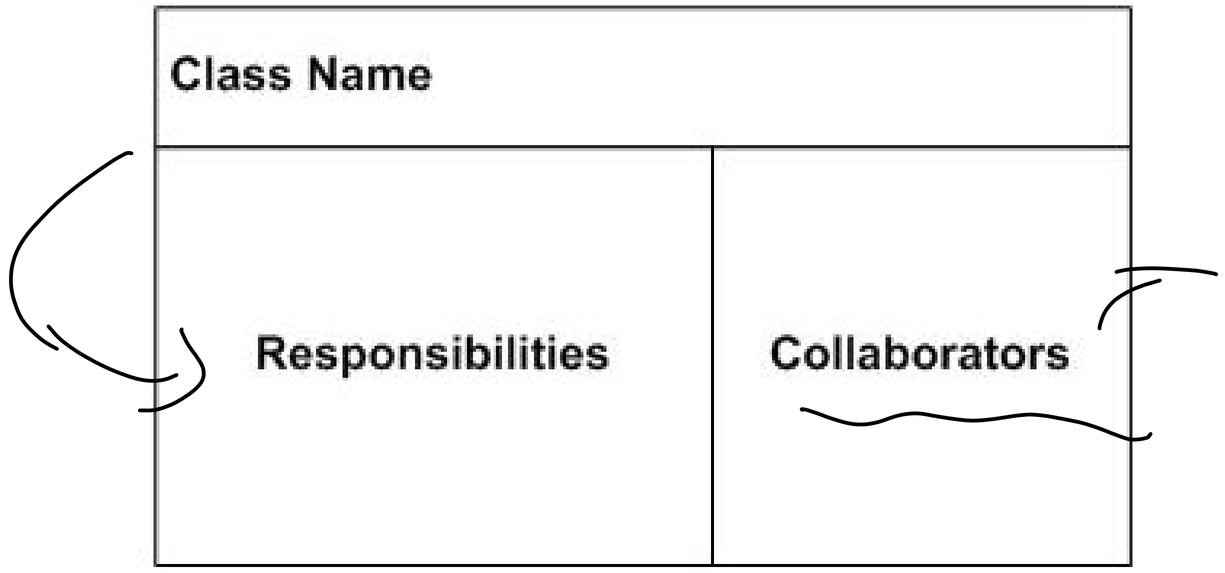
CRC Cards:

The card is partitioned into three areas:

I. On top of the card, the class name
II. On the left, the responsibilities of the class
III. On the right, collaborators (other classes) with which this class interacts to fulfill its responsibilities.

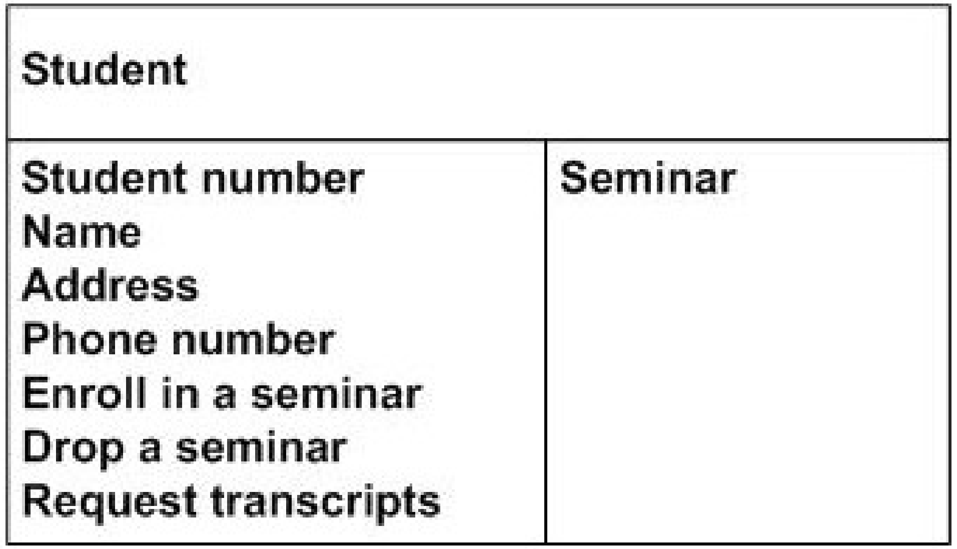
EXTREME PROGRAMMING (XP)

CRC Cards:

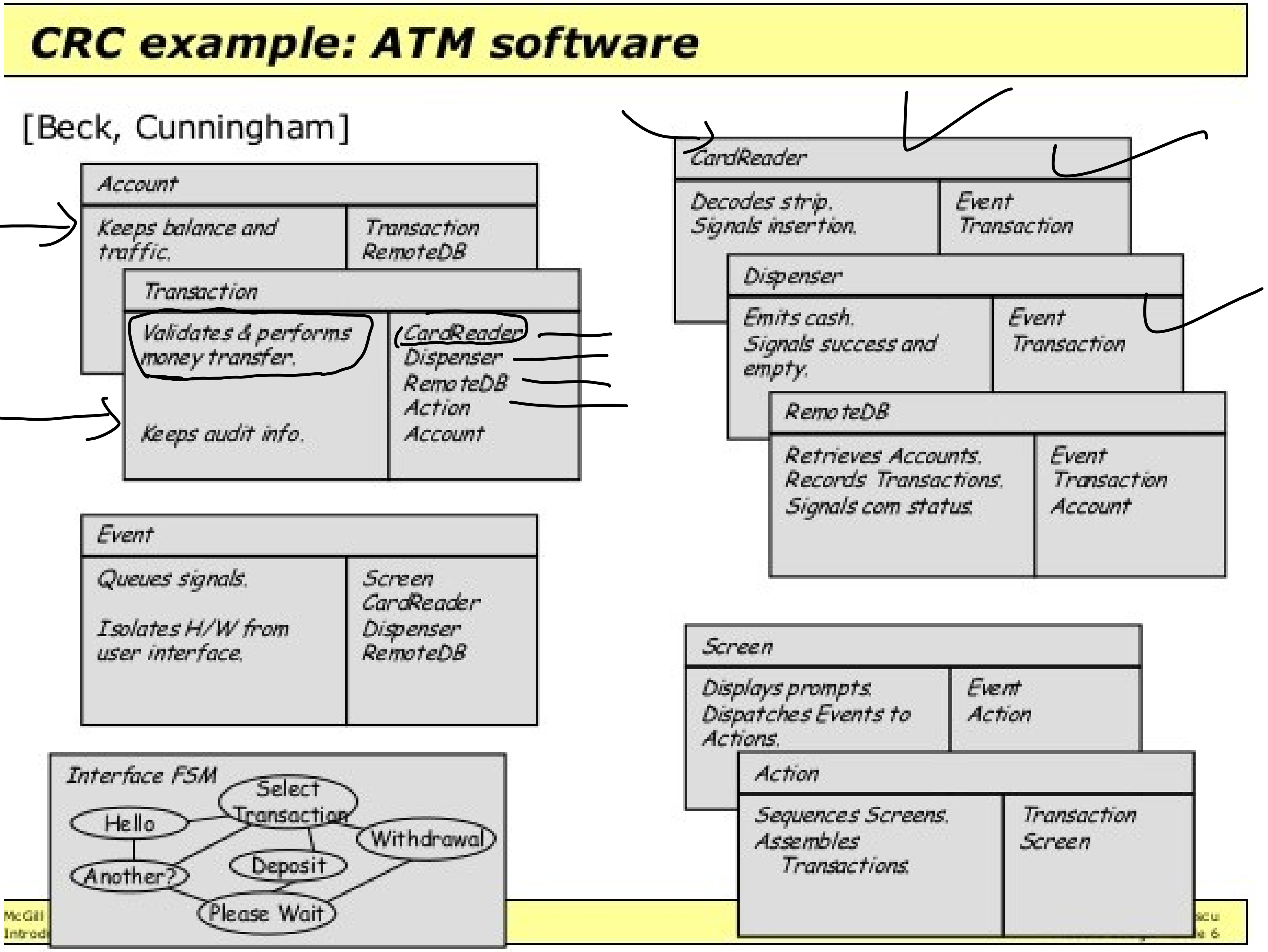


EXTREME PROGRAMMING (XP)

CRC Cards



Student	
Student number Name Address Phone number Enroll in a seminar Drop a seminar Request transcripts	Seminar



CRC example: ATM software

[Beck, Cunningham]

Account	
Keeps balance and traffic.	Transaction RemoteDB

Transaction	
Validates & performs money transfer. Keeps audit info.	CardReader Dispenser RemoteDB Action Account

Event	
Queues signals. Isolates H/W from user interface.	Screen CardReader Dispenser RemoteDB

CardReader	
Decodes strip. Signals insertion.	Event Transaction

Dispenser	
Emits cash. Signals success and empty.	Event Transaction

RemoteDB	
Retrieves Accounts. Records Transactions. Signals com status.	Event Transaction Account

Screen	
Displays prompts. Dispatches Events to Actions.	Event Action

Action	
Sequences Screens. Assembles Transactions.	Transaction Screen

Interface FSM

Tm

EXTREME PROGRAMMING (XP)

XP Coding

- Recommends the construction of a unit test for a store *before* coding commences
- Encourages pair programming

XP Testing

- All unit tests are executed daily
- Acceptance tests are defined by the customer and executed to assess customer visible functionality

TESTING IN XP

- Test-first development.
- Incremental test development from scenarios.
- User involvement in test development and validation.

TASK CARDS FOR DOCUMENT DOWNLOADING

Task 1: Implement principal workflow

Task 2: Implement article catalog and selection

Task 3: Implement payment collection

Payment may be made in 3 different ways. The user selects which way they wish to pay. If the user has a library subscription, then they can input the subscriber key which should be checked by the system. Alternatively, they can input an organisational account number. If this is valid, a debit of the cost of the article is posted to this account. Finally, they may input a 16 digit credit card number and expiry date. This should be checked for validity and, if valid a debit is posted to that credit card account.

TEST CASE DESCRIPTION

[illegible]

Test 4: Test credit card validity

Input:
A string representing the credit card number and two integers representing the month and year when the card expires
Tests:
Check that all bytes in the string are digits
Check that the month lies between 1 and 12 and the year is greater than or equal to the current year .
Using the first 4 digits of the credit card number , check that the card issuer is valid by looking up the card issuer table. Check credit card validity by submitting the card number and expiry date information to the card issuer
Output:
OK or error message indicating that the card is invalid

Input

Output

Expected Output

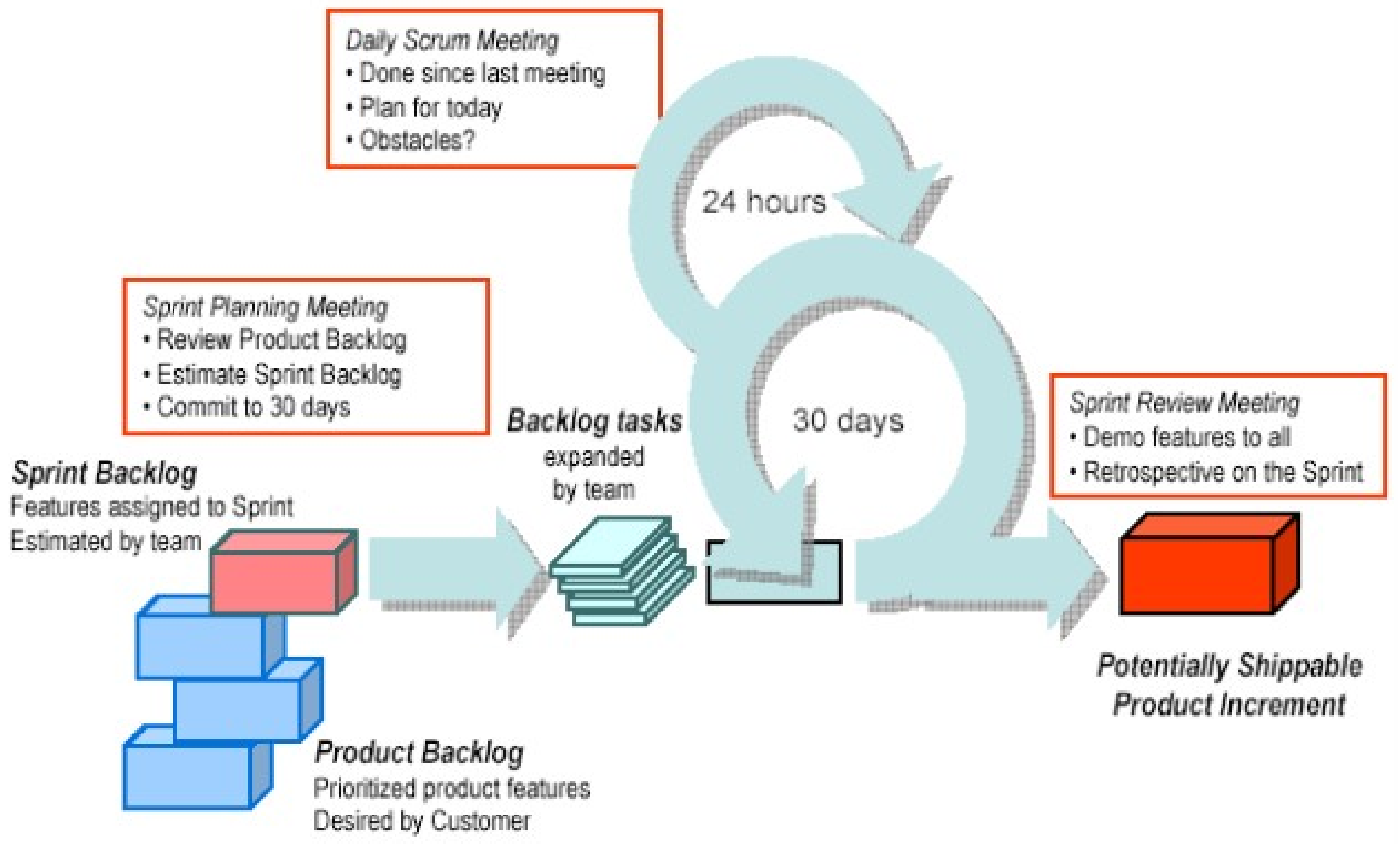
SIGNIFICANCE OF TEST-FIRST DEVELOPMENT

- Writing tests before code clarifies the requirements to be implemented.
- Tests are written as programs rather than data so that they can be executed automatically. The test includes a check that it has executed correctly.
- All previous and new tests are automatically run when new functionality is added. Thus checking that the new functionality has not introduced errors.

SCRUM

- Scrum is an Agile framework for completing complex projects.
- Scrum originally was formalized for software development projects, but it works well for any complex, innovative scope of work.
- Scrum is a team-based approach, to iteratively, incrementally develop systems and products.
- when requirements are rapidly changing .

HOW DOES SCRUM WORK?



How Scrum work

USER STORIES

User Story capture 3 important items

- Who
- What
- Why

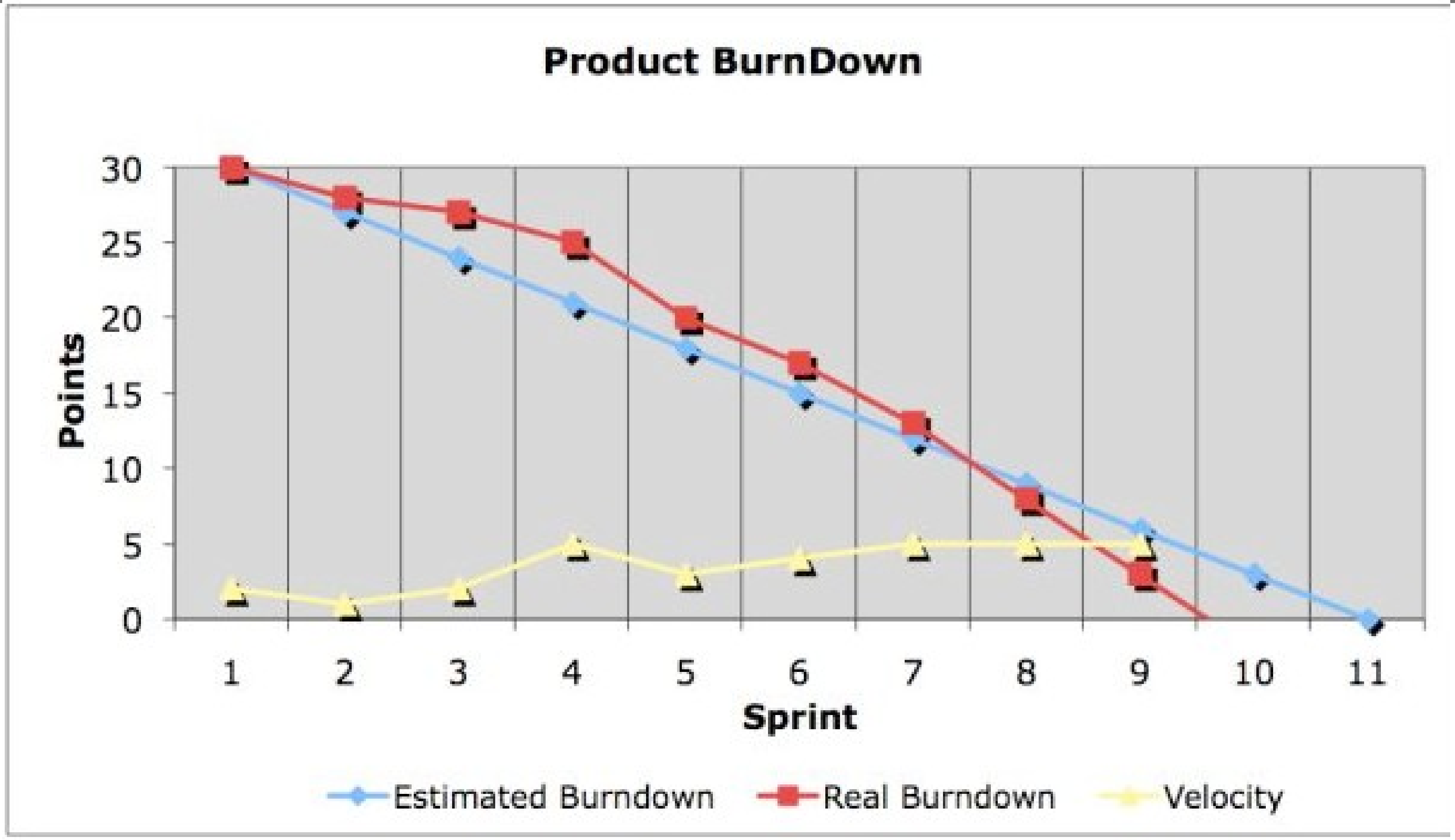
User Story Format

As a (user or type of user)

I want a (some goal or what)

So that (I can achieve some value or why)

BURNDOWN CHART



Simple Burndown Chart

DIFFERENCE BETWEEN XP AND SCRUM

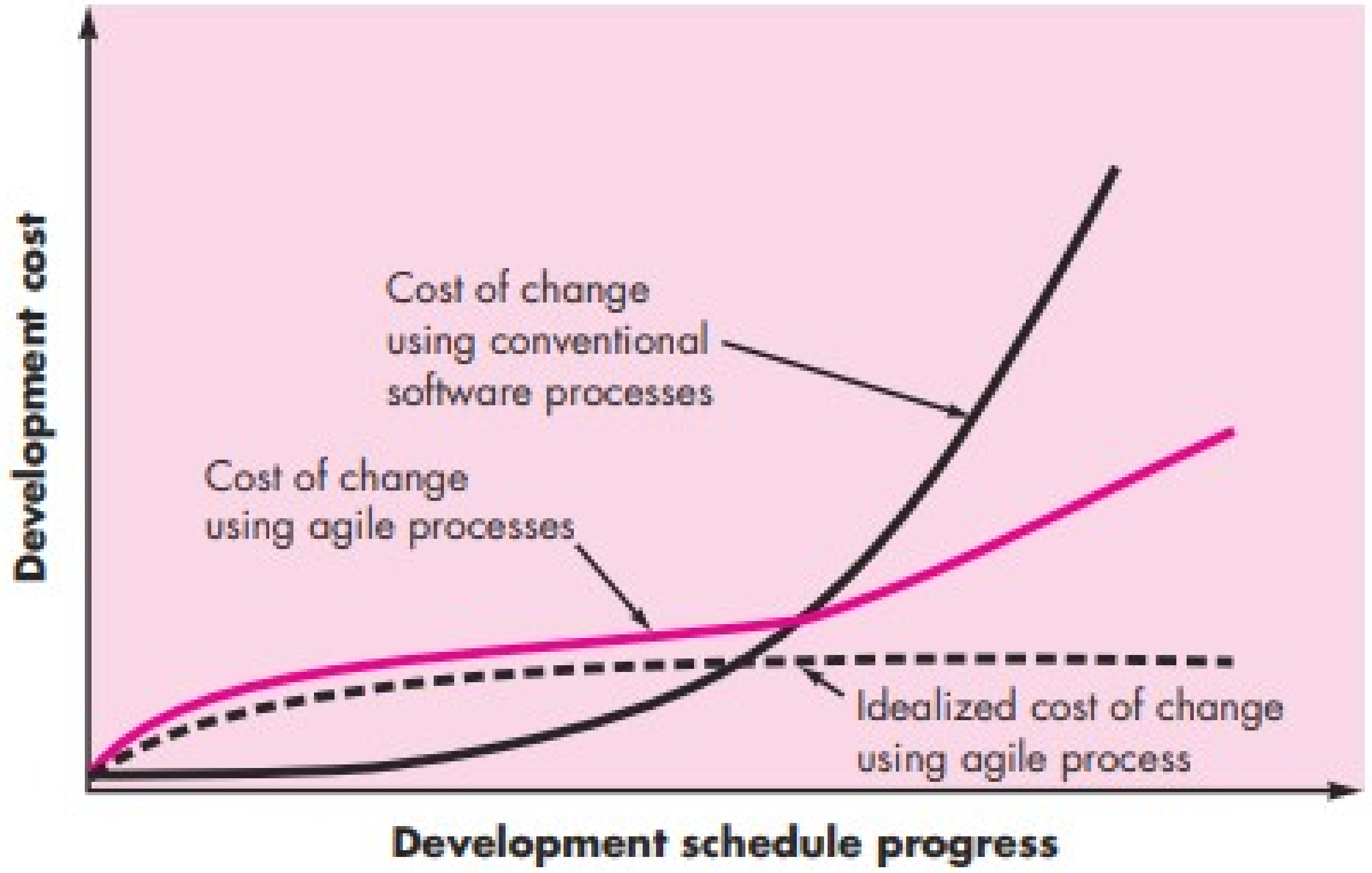
XP Iterations (1-2 Weeks)

Scrum Iterations (3-4Weeks)

Scrum teams (do not allow changes into their sprints).

XP team (allow changes)

COST OF CHANGE IN AGILE



REQUIREMENT ENGINEERING

WHAT IS REQUIREMENT?

What is requirement?

- **The descriptions of what the system should do**
  - **services that it provides and the constraints**

TYPES OF REQUIREMENTS

- **Functional requirements:**
    - statement of **services**
    - how system **reacts** to input
    - how system **behaves** in particular situation
- **Non-functional requirements:**
    - constraints on services **(timing, quality, security etc.)**

- Domain requirements
- Inverse requirements
- Design and implementation constraints

Why Requirement Engineering:

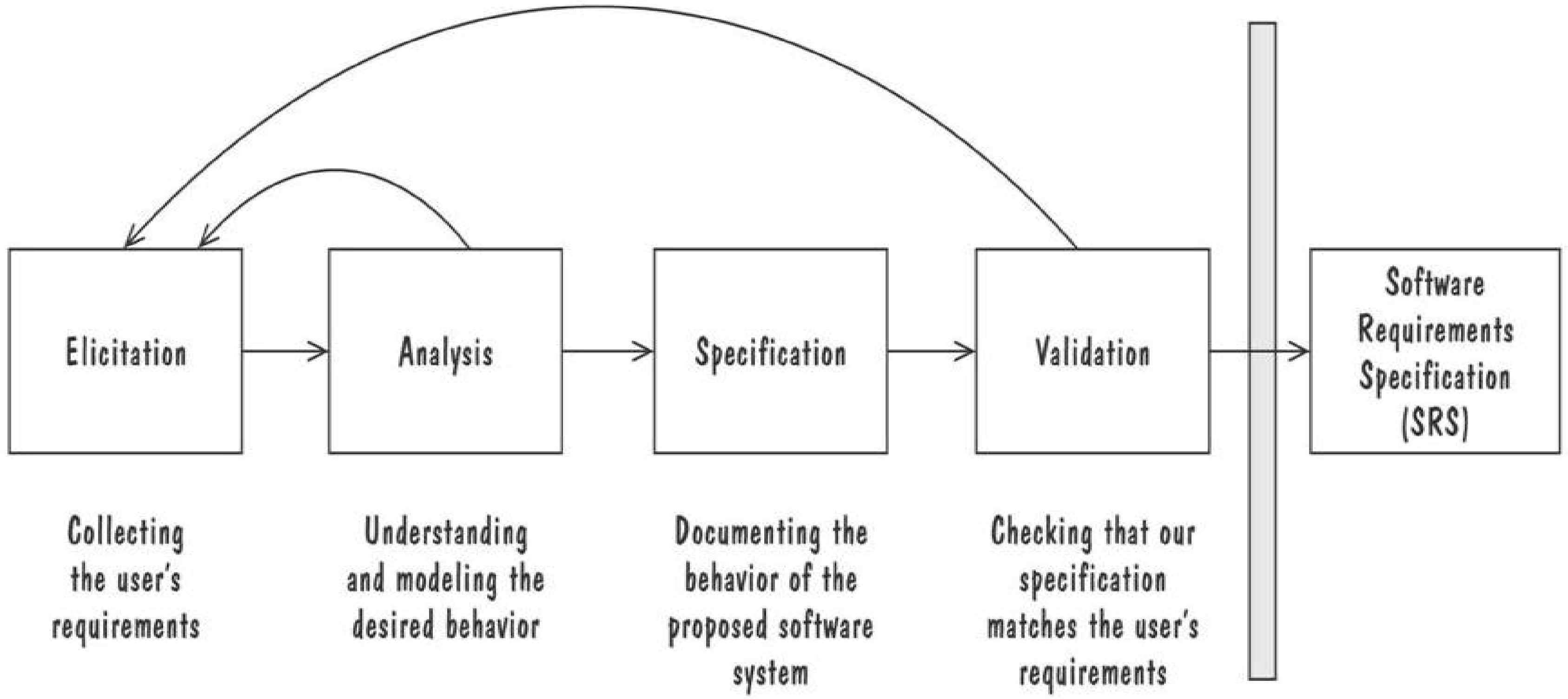
- A study based on 340 companies in Austria, more than two thirds consider the SRS as the major problem in development process (1995)
- A study on Web applications, 16% systems fully meet their requirement while 53% deployed systems do not (Cutter Consortium, 2000)

Why Requirement Engineering:

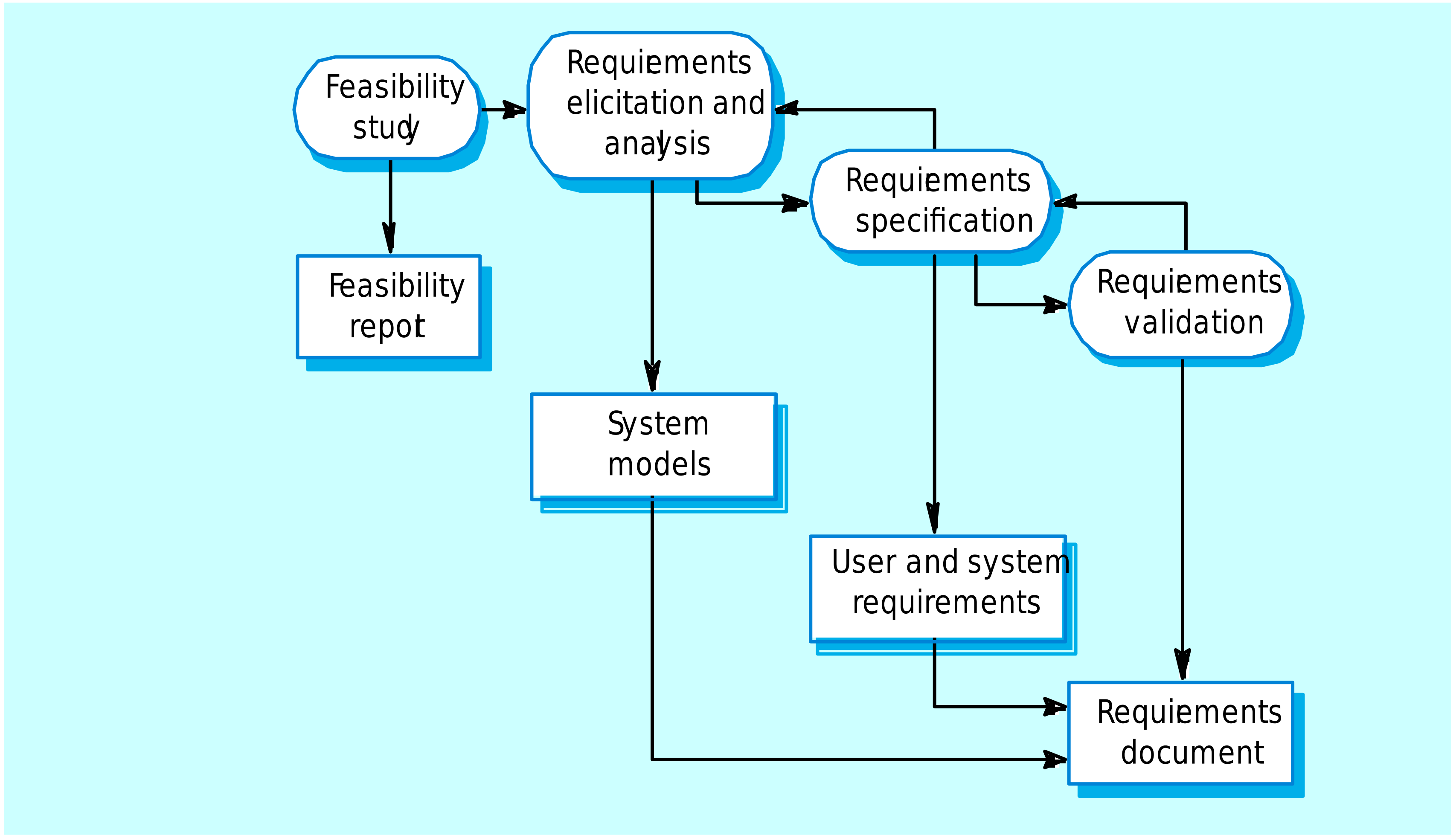
- **A study among 8000 projects, 30% of projects fail before completion & almost half do not meet customer requirements (Standish group, 1994)**
  - **Unclear objectives, unrealistic schedules & expectations, poor user participation**

THE REQUIREMENTS PROCESS (PROCESS FOR CAPTURING REQUIREMENTS)

- Performed by the req. analyst or system analyst
- The final outcome is a Software Requirements Specification (SRS) document


Elicitation
Analysis
Specification
Validation
Software Requirements Specification (SRS)
Collecting the user's requirements
Understanding and modeling the desired behavior
Documenting the behavior of the proposed software system
Checking that our specification matches the user's requirements

REQUIREMENT ENGINEERING PROCESS



REQUIREMENTS ELICITATION
IDENTIFY SOURCES OF REQUIREMENTS

- Interviewing stakeholders
- Reviewing available documentations
- Observing the current system (if one exists)

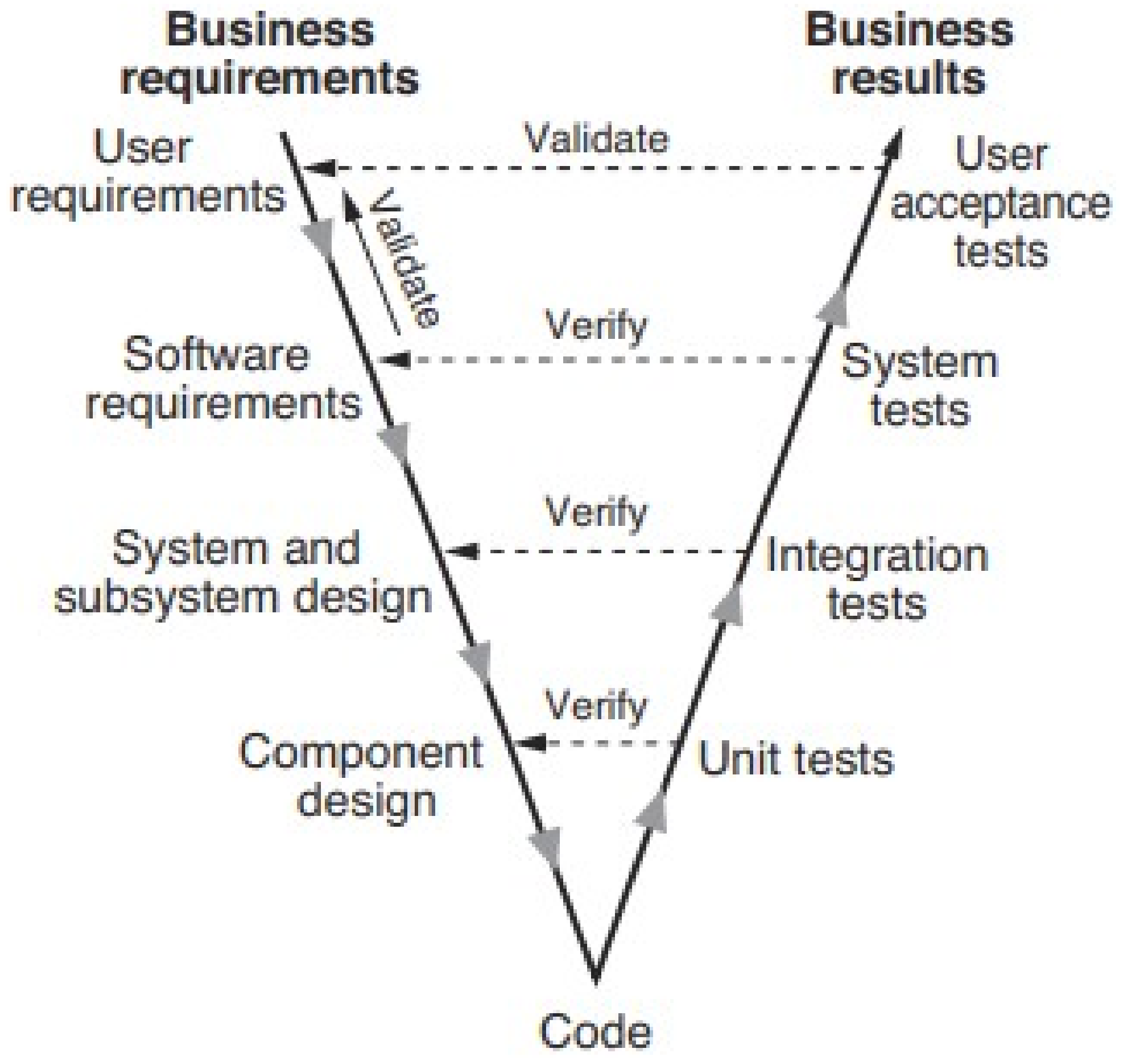
REQUIREMENTS ELICITATION
STAKEHOLDERS

- Clients: pay for the software to be developed
- Users: use the system
- Domain experts: familiar with the problem that the software must automate

REQUIREMENTS ELICITATION
STAKEHOLDER LIST

- Auditor
- Buyer
- Clerical user
- Customer service analyst
- Database administrator
- Financial expert
- Sales specialist
- Software Architect
- Network Administrator
- Usability specialist
- Security Specialist

How Requirements are Verified and Validated



VERIFICATION

- Verification addresses the concern: "Are you building it right?"
- Ensures that the software system meets all the functionality.
- Verification takes place first and includes the checking for documentation, code, etc.
- Done by developers.
- It has static activities, as it includes collecting reviews, walkthroughs, and inspections to verify a software.
- It is an objective process and no subjective decision should be needed to verify a software.

VALIDATION

- Validation addresses the concern: "Are you building the right thing?"
- Ensures that the functionalities meet the intended behavior.
- Validation occurs after verification and mainly involves the checking of the overall product.
- Done by testers.
- It has dynamic activities, as it includes executing the software against the requirements.
- It is a subjective process and involves subjective decisions on how well a software works.

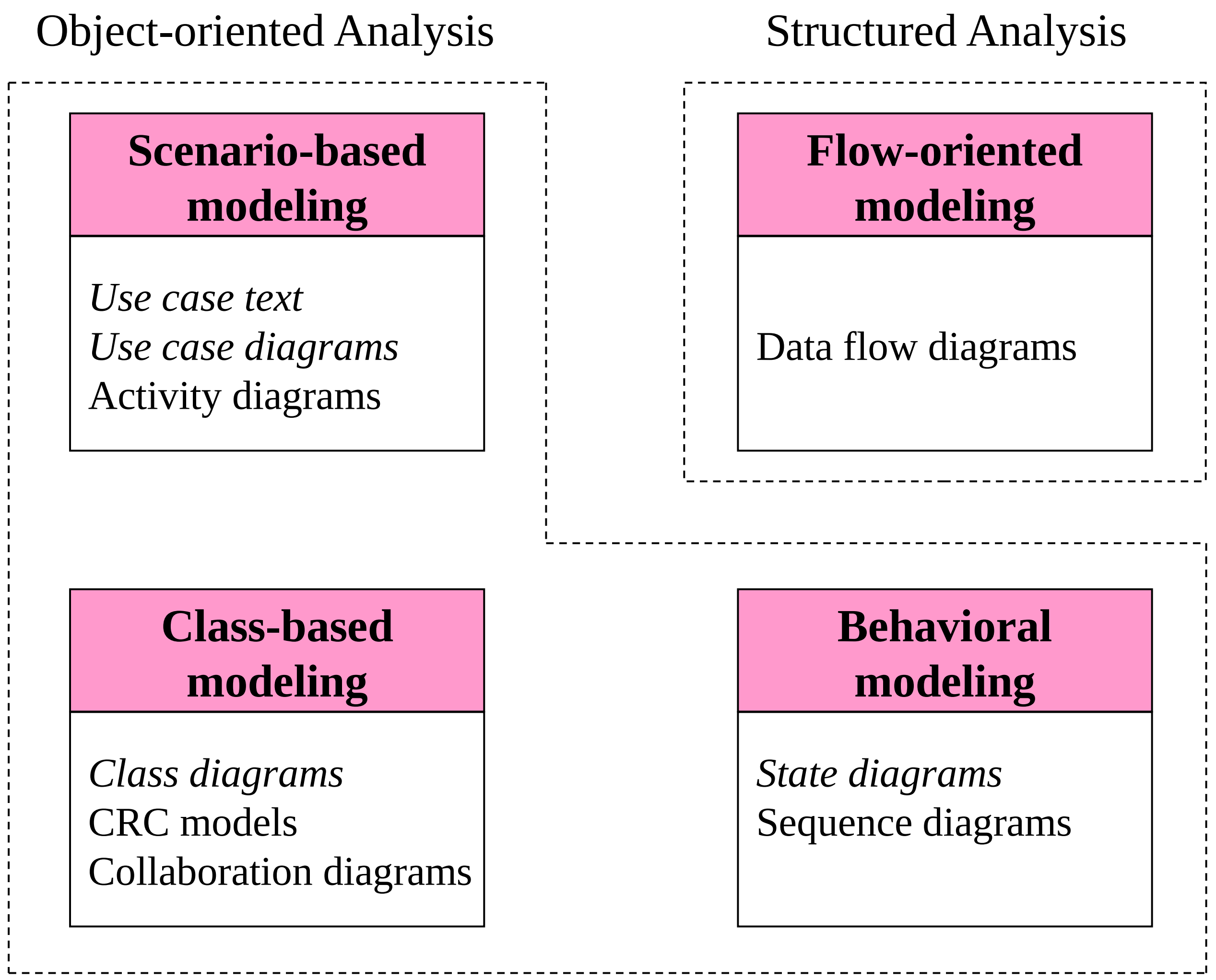
REQUIREMENT ANALYSIS

ANALYSIS MODELING

ANALYSIS MODEL

- Analysis results in requirements models.
- Requirements models (also referred to as analysis models) are user requirements represented by diagrams.

ELEMENTS OF THE ANALYSIS MODEL

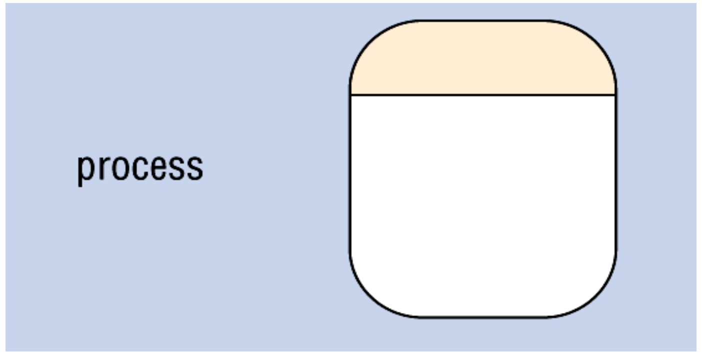


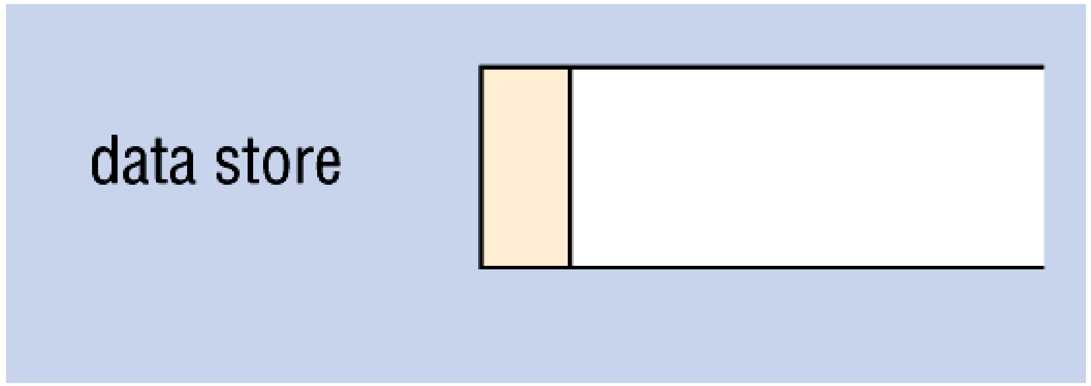
FLOW-ORIENTED MODELING

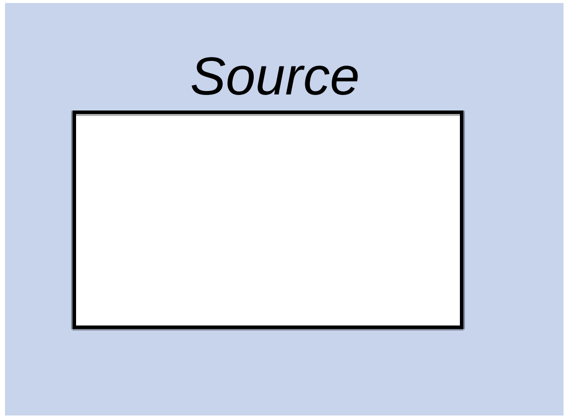
WHAT IS A DATA FLOW DIAGRAM?

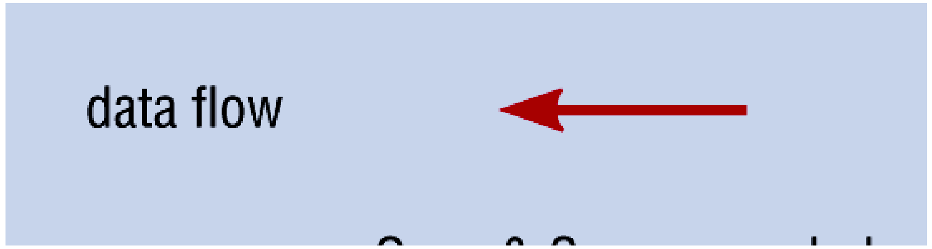
- A data flow diagram (DFD) is a graphical tool that allows system analysts (and system users) to depict the flow of data in an information system.

DATA FLOW DIAGRAM SYMBOLS





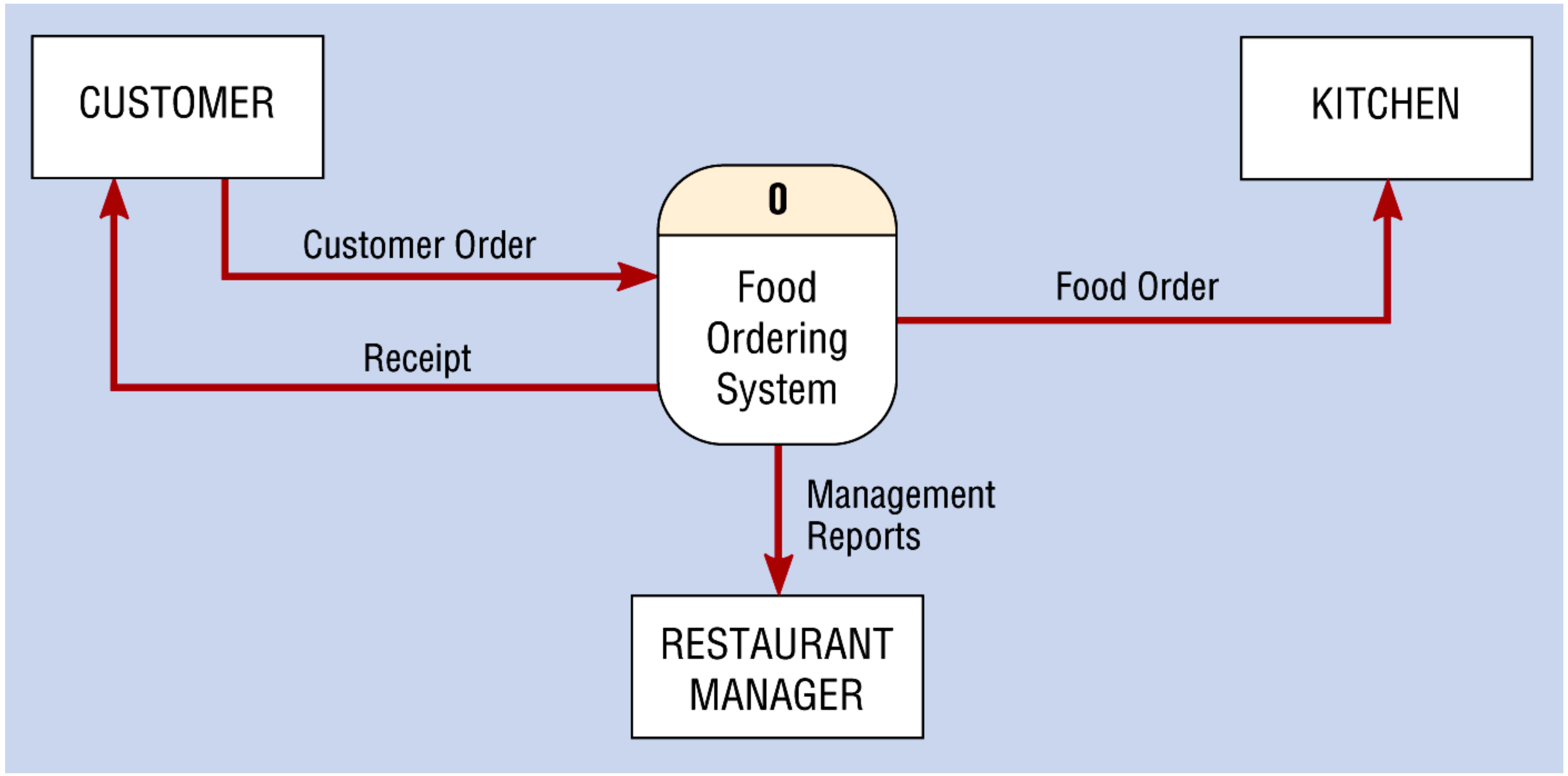




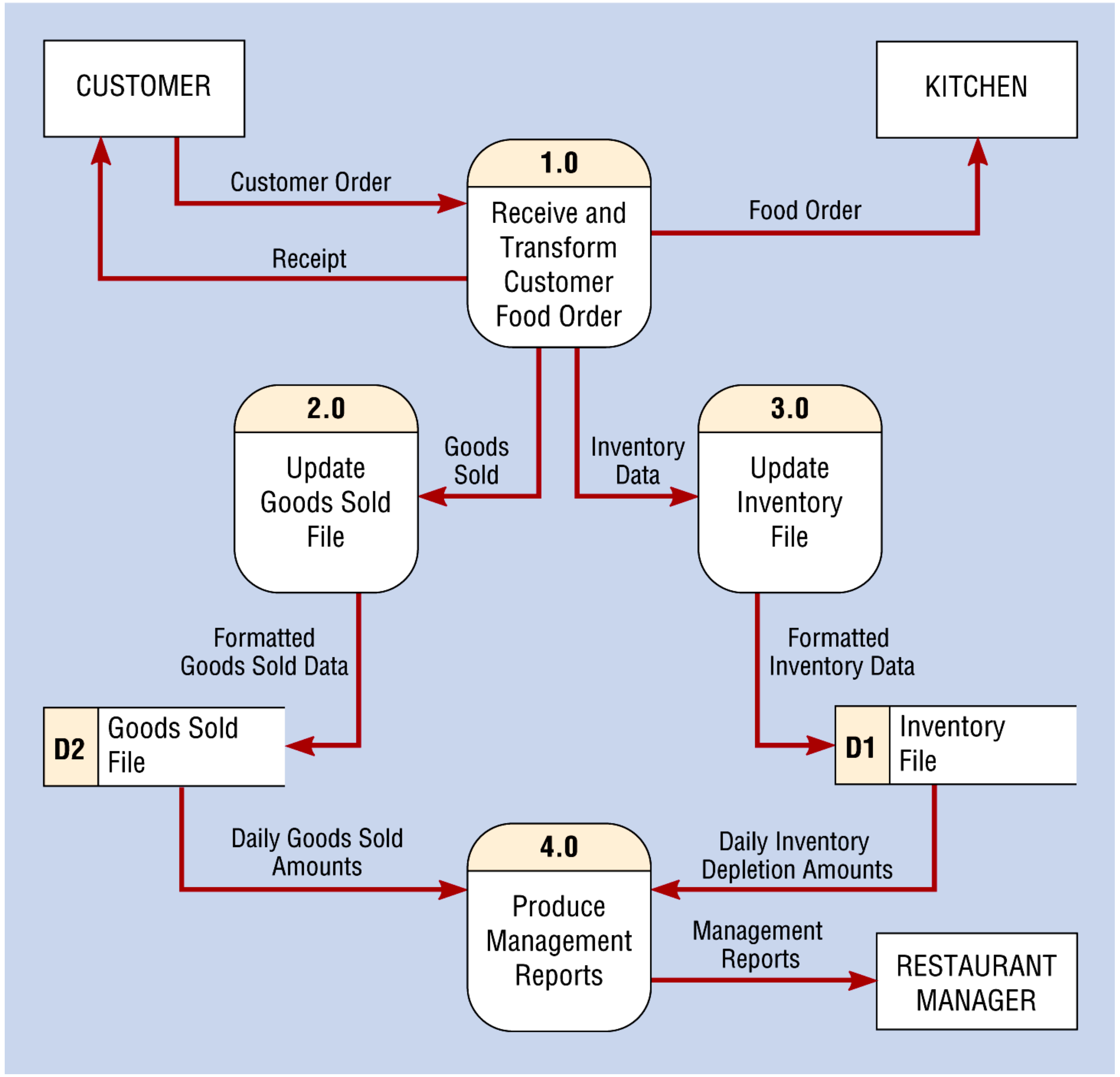
STEPS IN BUILDING DFDS

- Build the context diagram
- Create DFD fragments
- Organize DFD fragments into level 0
- Decompose level 0 DFDs as

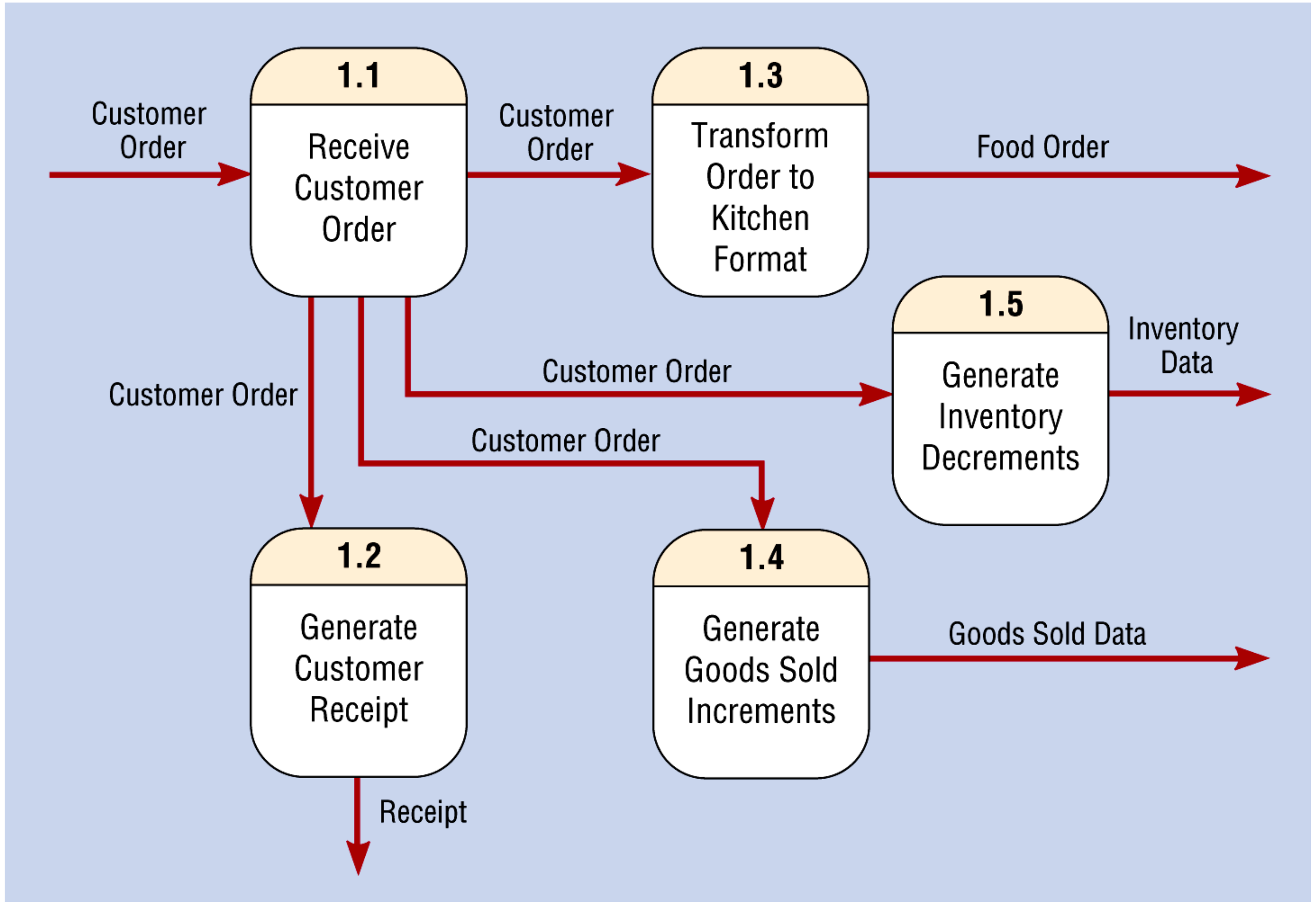
CONTEXT DIAGRAM OF FOOD ORDERING SYSTEM



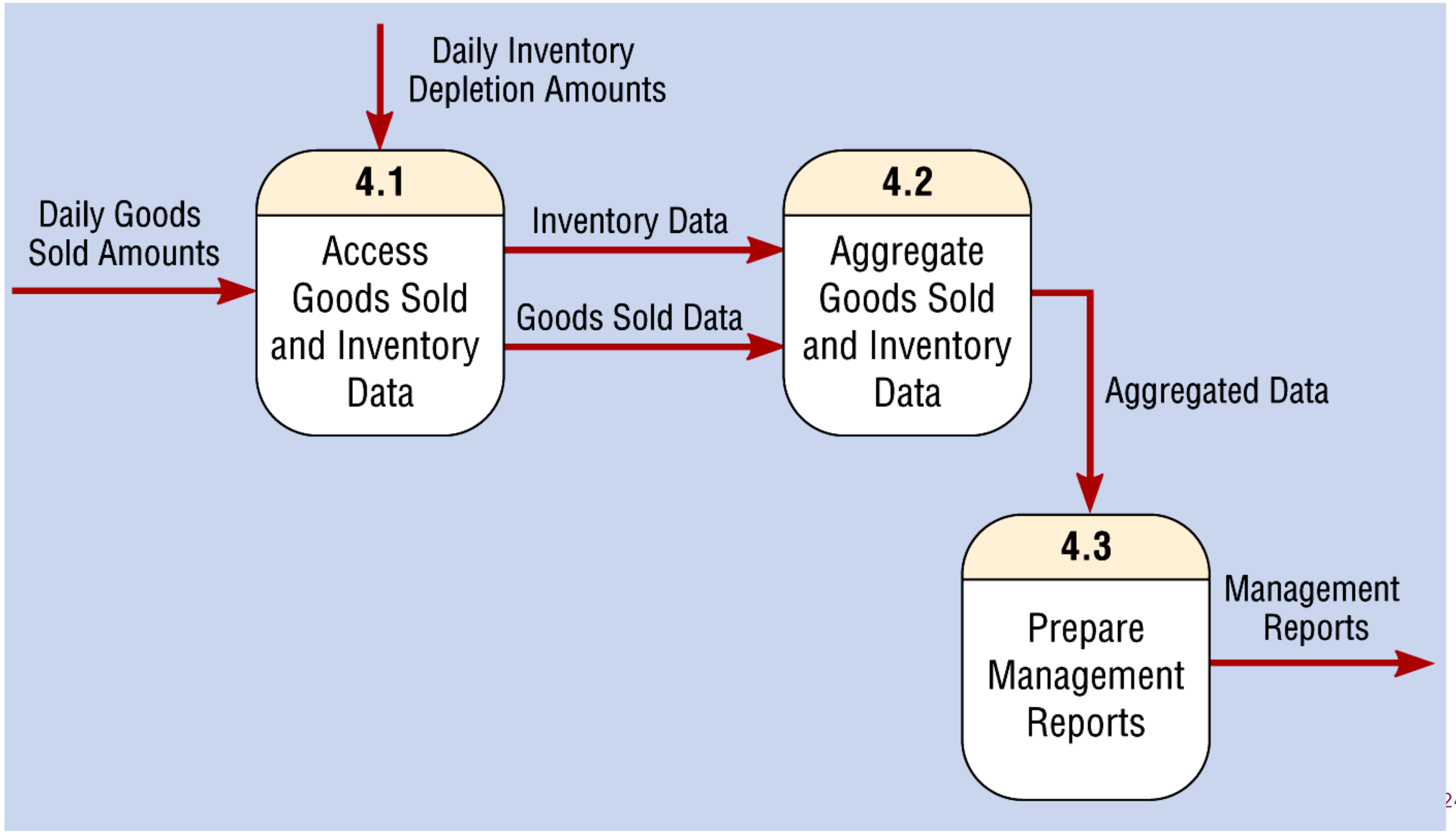
LEVEL-0 DFD OF FOOD ORDERING SYSTEM



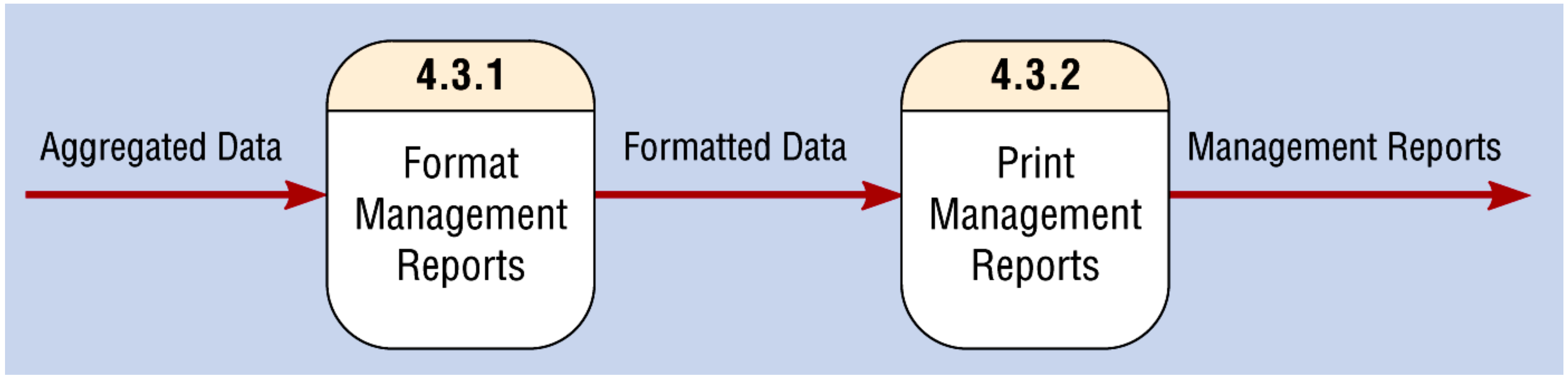
LEVEL-1 DIAGRAM SHOWING DECOMPOSITION OF PROCESS 1.0 FROM THE LEVEL-0 DIAGRAM



LEVEL-1 DIAGRAM SHOWING THE DECOMPOSITION OF PROCESS 4.0 FROM THE LEVEL-0 DIAGRAM

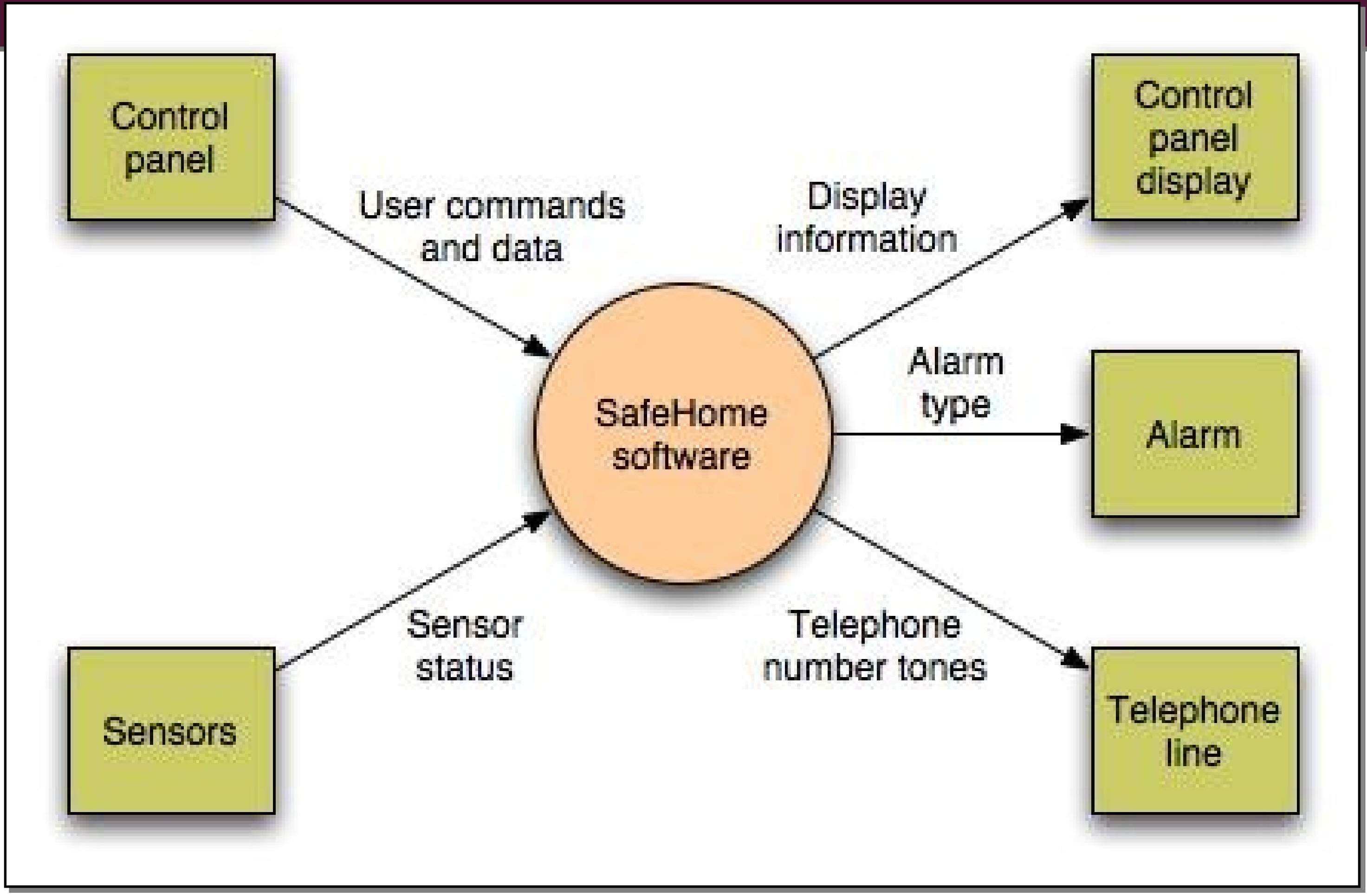

Daily Inventory Depletion Amounts
Daily Goods Sold Amounts
4.1
Access Goods Sold and Inventory Data
Inventory Data
Goods Sold Data
4.2
Aggregate Goods Sold and Inventory Data
Aggregated Data
4.3
Prepare Management Reports
Management Reports

LEVEL-2 DIAGRAM SHOWING THE DECOMPOSITION OF PROCESS 4.3 FROM THE LEVEL-1 DIAGRAM FOR PROCESS 4.0

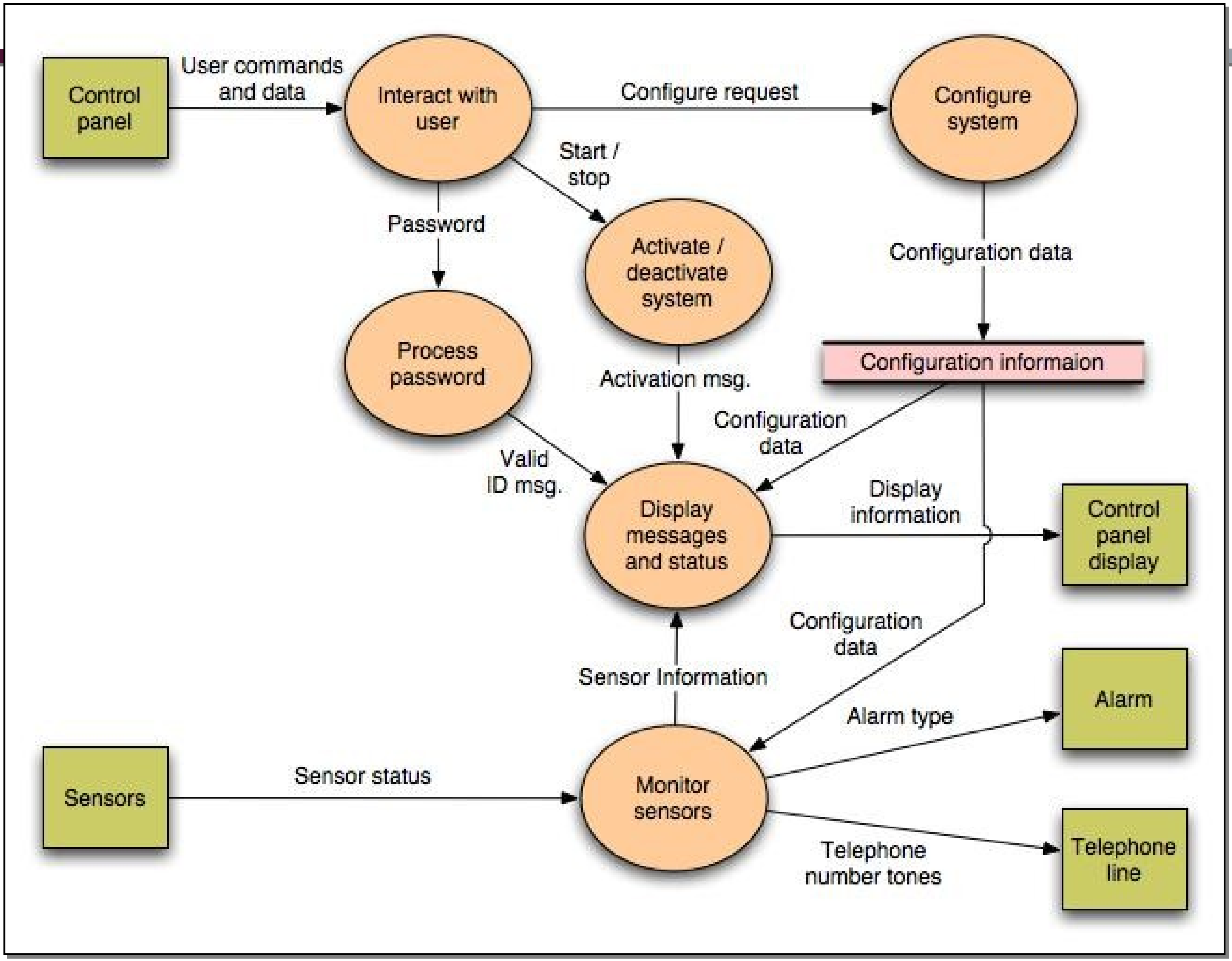


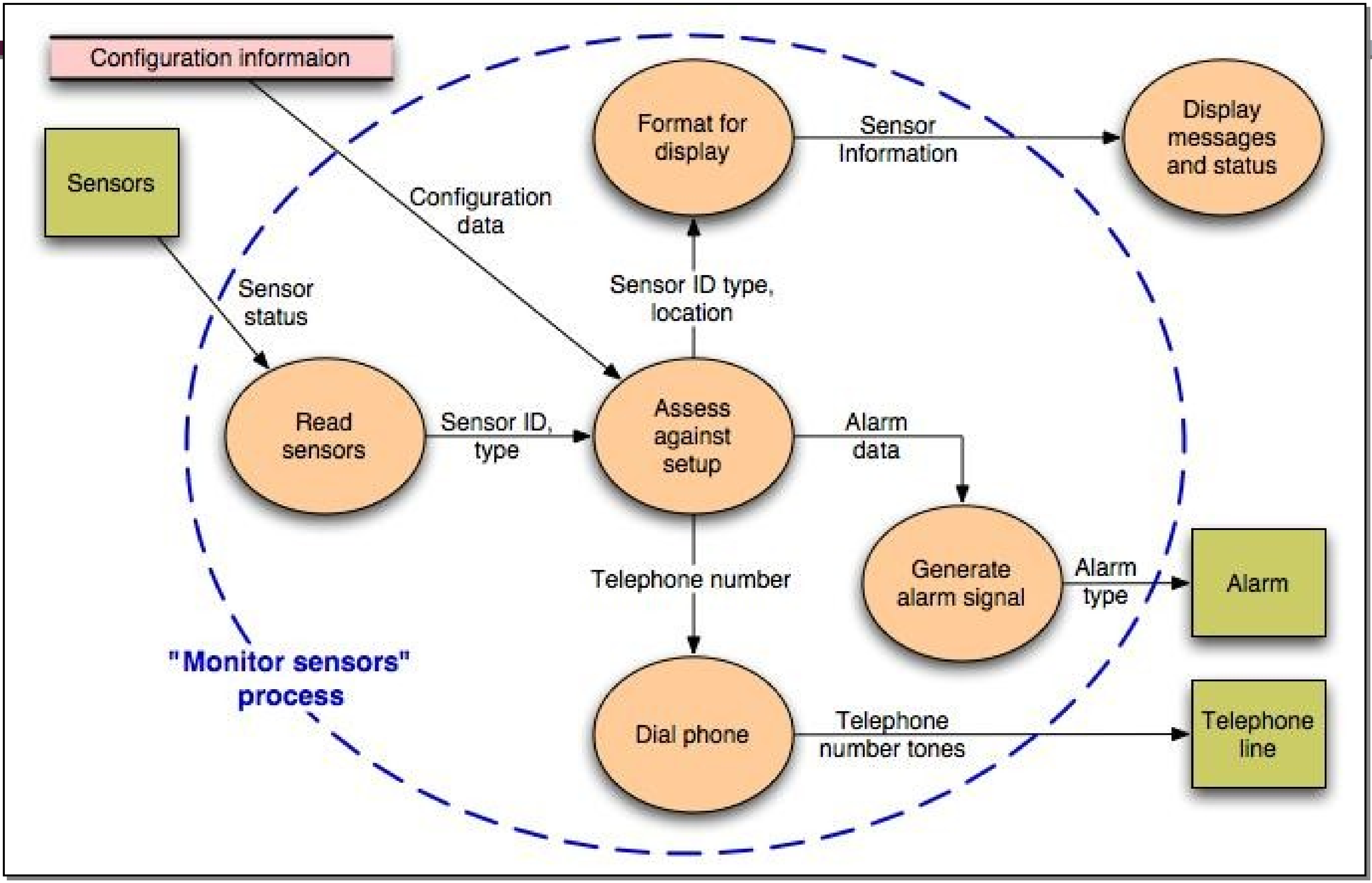
DATA FLOW DIAGRAM OF SAFE HOME SYSTEM

DATA FLOW DIAGRAM



Context-level DFD for *SafeHome* security function


Control panel
User commands and data
Interact with user
Configure request
Configure system
Start / stop
Password
Activate / deactivate system
Configuration data
Process password
Configuration informaion
Activation msg.
Configuration data
Valid ID msg.
Display messages and status
Display information
Control panel display
Configuration data
Sensor Information
Alarm
Alarm type
Sensors
Sensor status
Monitor sensors
Telephone line
Telephone number tones



Level 2 DFD that refines the monitor sensors process

HAVE A GOOD DAY!

Waterfall Model:

2 Months / Customer = ??

① Sequential Model

② Clear Steps.

① ReQ - - - ✓ SRS 1 Month

② Design ✓ → Design Draft 1 Month

③ Coding ✓

④ Testing

Cost

changes

No Customer Involvement

X

WM

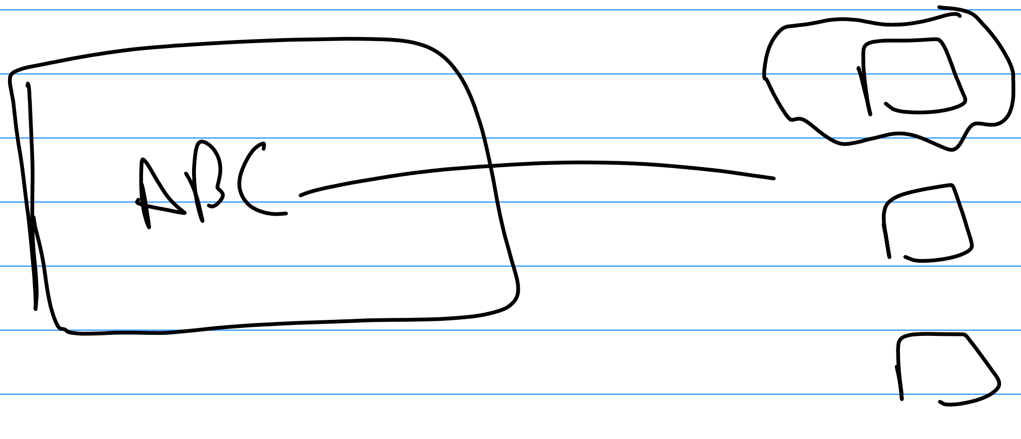
① If ReQ are Clear = ✓

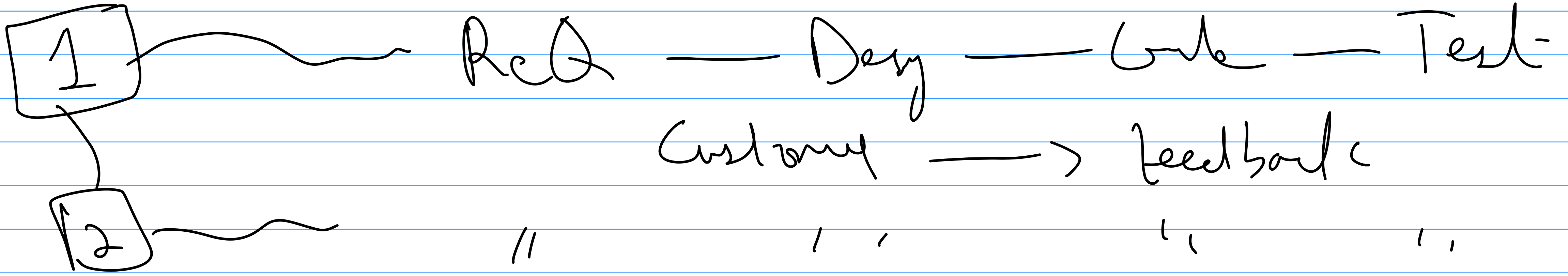
② Scope of project is smaller = ✓

<u>ReQuirement Not Clear</u>

Incremental Approach

Approach ⟶ in the form
increments



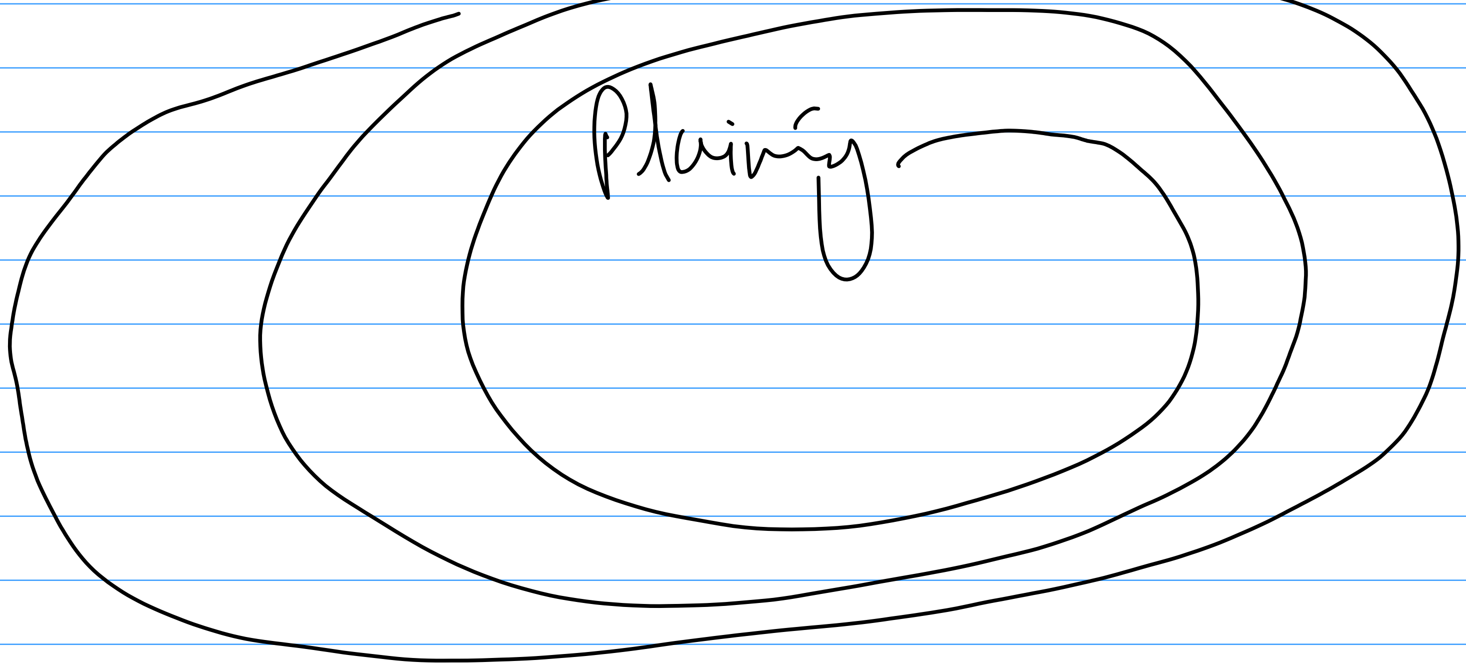


Waterfall	Incremental Approach
Overall Risk	Lower Risk of
✓ Project Failure	Project Failure

Spiral Model

→ Incremental

Planning



Risk Management

→ Project Managemt

Software

- Health Critical
  - Autonomy Car
  - Body Area Network (
  - Spiral Model
- Mission Critical
  - Bank,
  - Web Application

Formal Methods in SE

Natural Language

ReQ Engg $\longrightarrow$ Formal

(Mathematical Notation

Discrete Structures + Automata

$2 + 2 = 4$