MODULE 1:

SOFTWARE PROCESS MODEL

Software:

computer program and associated documentation

software encompasses:

1) Instruction (computer programs)
2) Data structures => enable the program to adequately store and manipulate
3) Documentation => describes operation and use of program

Software Products

Generic product: Stand-alone system that are marketed and sold to any customes

Eg: OS, PC software

customized product: sold only to specific customer.

Software cost:

Software engineering is concerned with cost-effective software development.

Attributes of Good Software:

=> should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.

Software Engineering :

Software engineering is an engineering discipline that is concerned with all aspects of software production.

Attributes :
Maintainability, Dependability, Security, Efficiency, Acceptability

Process Model

=> An abstract representation of process.

=> Software process Model is also called as Software development life cycle.

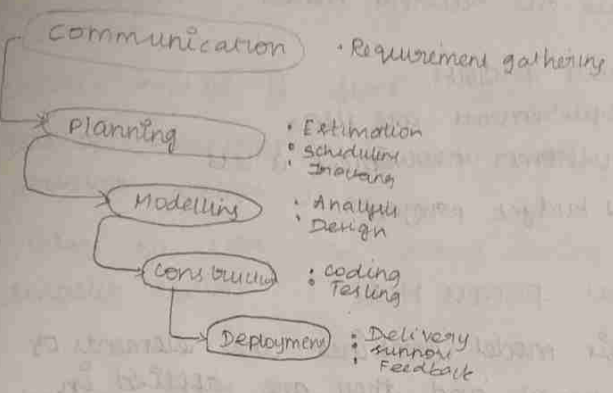
Different perceptive process model.

i) The Waterfall Modal

=> Traditional model

=> Only for small project

=> The waterfall is also called as Linear sequential model.



Alternate design.

9/8

Advantage of waterfall model:

⇒ The water fall model is simple and easy to understand, implement and use

⇒ All the requirements are known at the beginning of project, hence it is easy to manage

⇒ It avoids overlapping of phases because each phase is completed once.

Disadvantage.

⇒ This model is not good for complex and object oriented projects

⇒ This is a poor model for long projects

⇒ amount of risk is high.

When to use the waterfal model

⇒ For small projects
⇒ When requirements are clear
⇒ when customer involvement is less
⇒ For low budget project.

Incremental process Model

⇒ In this model combines the elements of waterfall model and they are applied in an iterative fashion

⇒ The first increment in this model is generally a core product.

⇒ Each increment builds the product and submit it to the customer for any suggested modification

⇒ The next increment implementation on the customer suggestions and add additional requirements in the previous increment

⇒ This process is repeated until the product finished.

Advantage:

⇒ This model is flexible because the cost of development is low and initial product deliver is faster

⇒ Easy to test and debug during the smaller project

Disadvantage

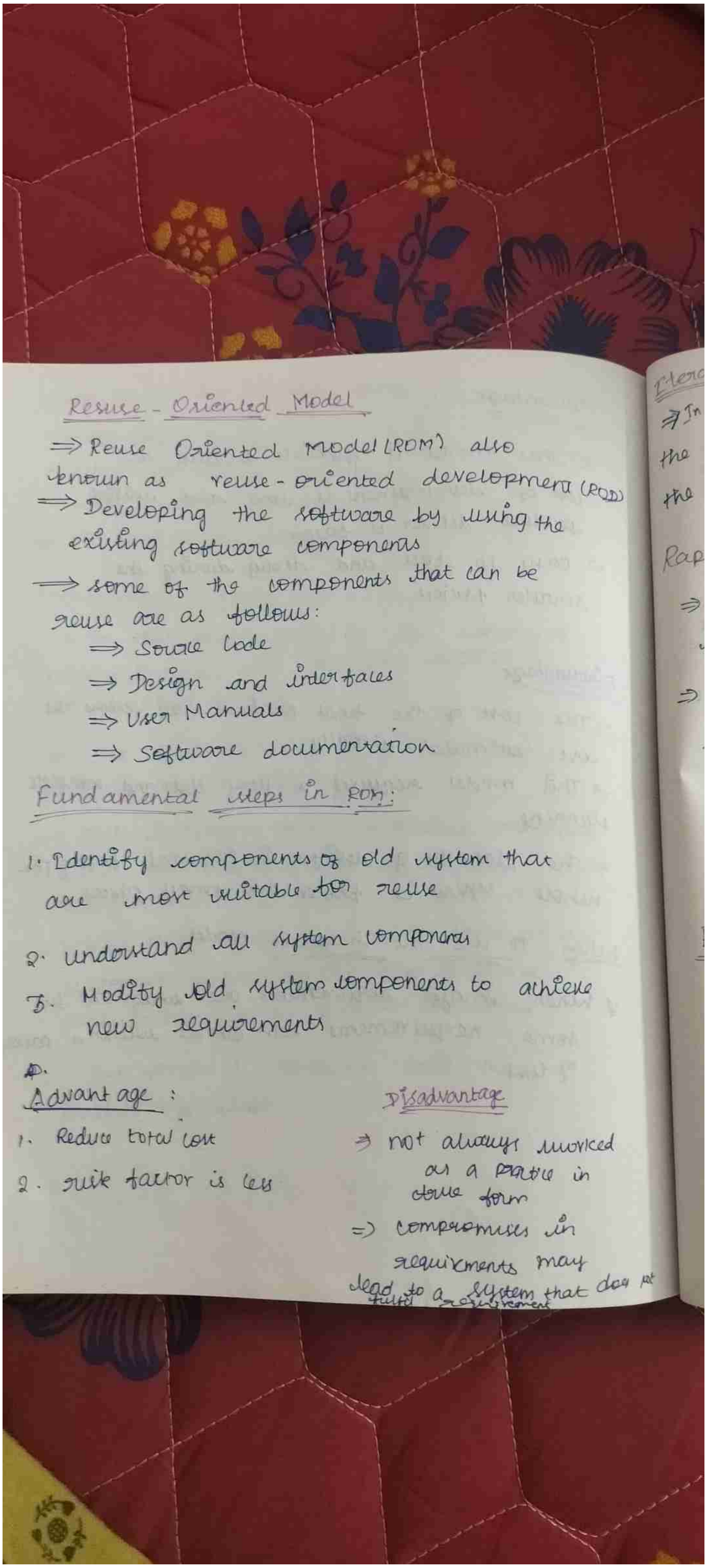
⇒ The cost of the final product may cross the cost estimated initially

⇒ This model requires a very clear and complete planning

⇒ The planning of design is required before the whole system is broken into small pieces

When to use the increment model

∅ When major requirements are understood but some requirements can evolve within a period of time



Reuse - Oriented Model

⇒ Reuse Oriented model (ROM) also known as reuse - oriented development (ROD)

⇒ Developing the software by using the existing software components

⇒ some of the components that can be reuse are as follows:

⇒ Source code

⇒ Design and interfaces

⇒ User Manuals

⇒ Software documentation

Fundamental steps in ROM:

1. Identify components of old system that are most suitable for reuse

2. understand all system components

3. Modify old system components to achieve new requirements

4.

Advantage:

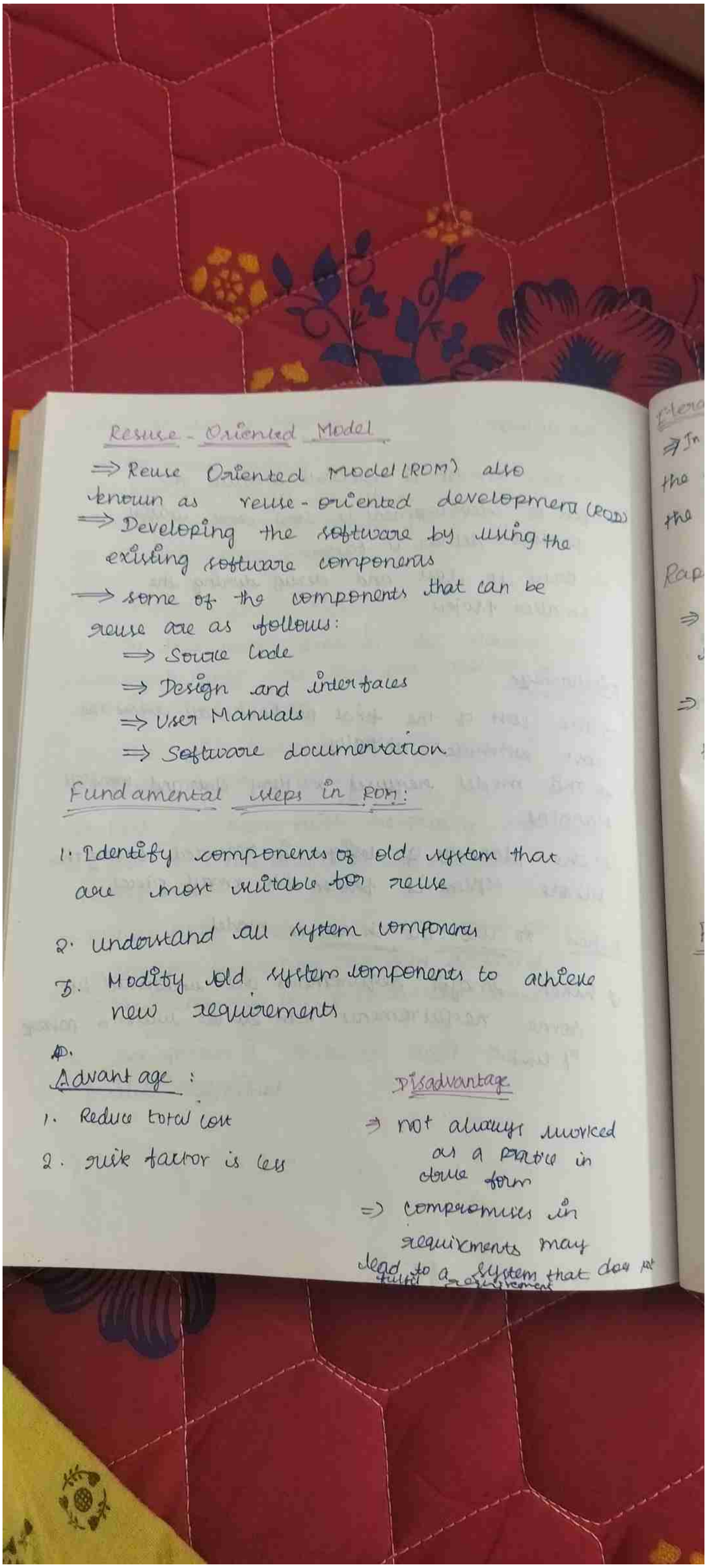
1. Reduce total cost

2. risk factor is less

Disadvantage

⇒ not always worked as a practice in true form

=) compromises in requirements may lead to a system that does not fulfil a requirement



Reuse - Oriented Model

⇒ Reuse Oriented model (ROM) also known as reuse - oriented development (ROD)

⇒ Developing the software by using the existing software components

⇒ some of the components that can be reuse are as follows:

⇒ Source Code

⇒ Design and interfaces

⇒ User Manuals

⇒ Software documentation

Fundamental steps in ROM:

1. Identify components of old system that are most suitable for reuse

2. understand all system components

3. Modify old system components to achieve new requirements

Advantage :

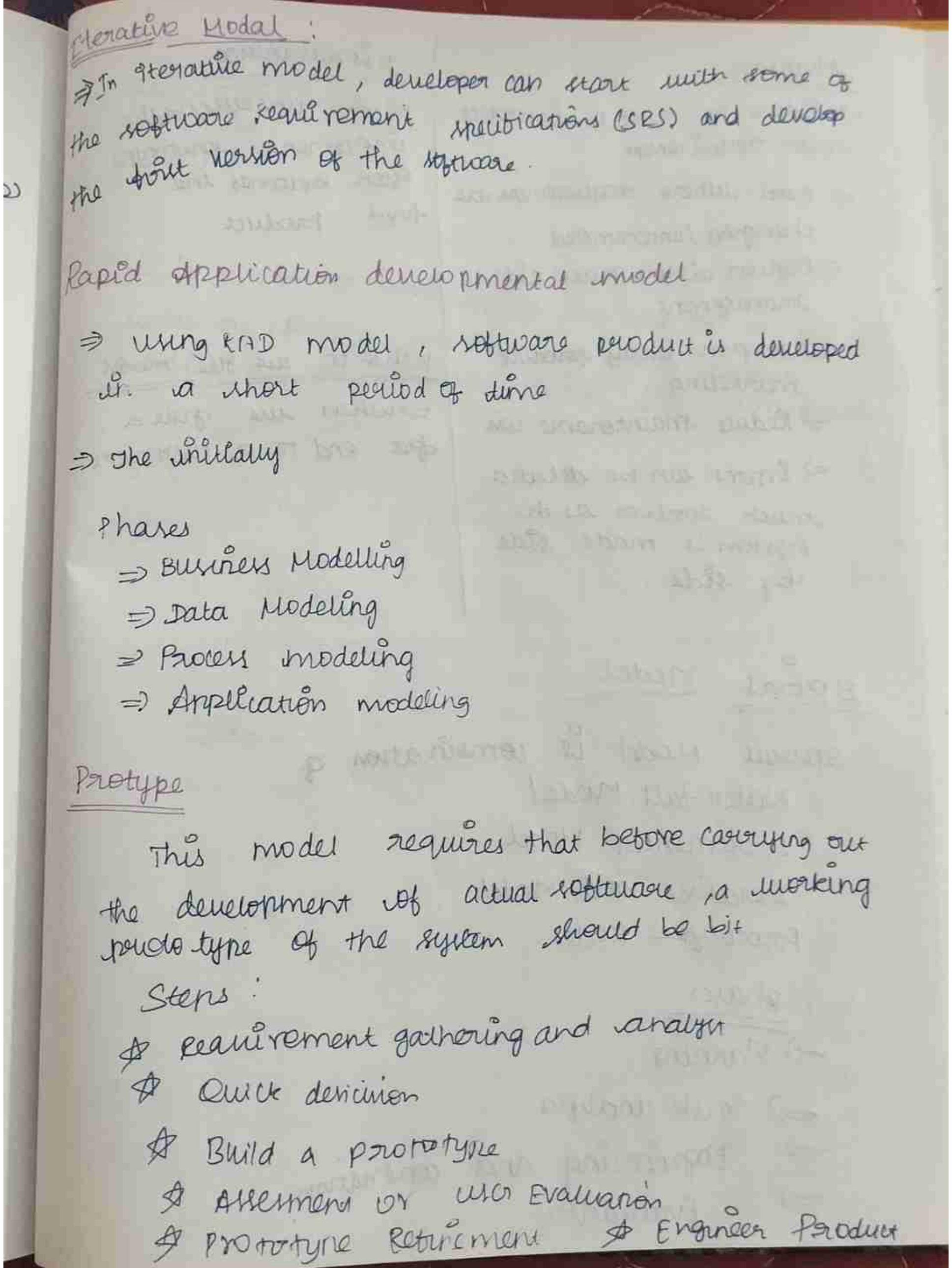
1. Reduce total cost

2. risk factor is less

Disadvantage

⇒ not always worked as a pratice in true form

⇒ compromises in requirements may lead to a system that does not fulfil a requirement



Iterative Modal :

⇒ In iterative model, developer can start with some of the software requirement specifications (SRS) and develop the first version of the software.

Rapid application developmental model

⇒ using RAD model, software product is developed in a short period of time

⇒ The initially

Phases

⇒ Business Modelling

⇒ Data Modeling

⇒ Process modeling

⇒ Application modeling

Protype

This model requires that before carrying out the development of actual software, a working proto type of the system should be bit

Steps :

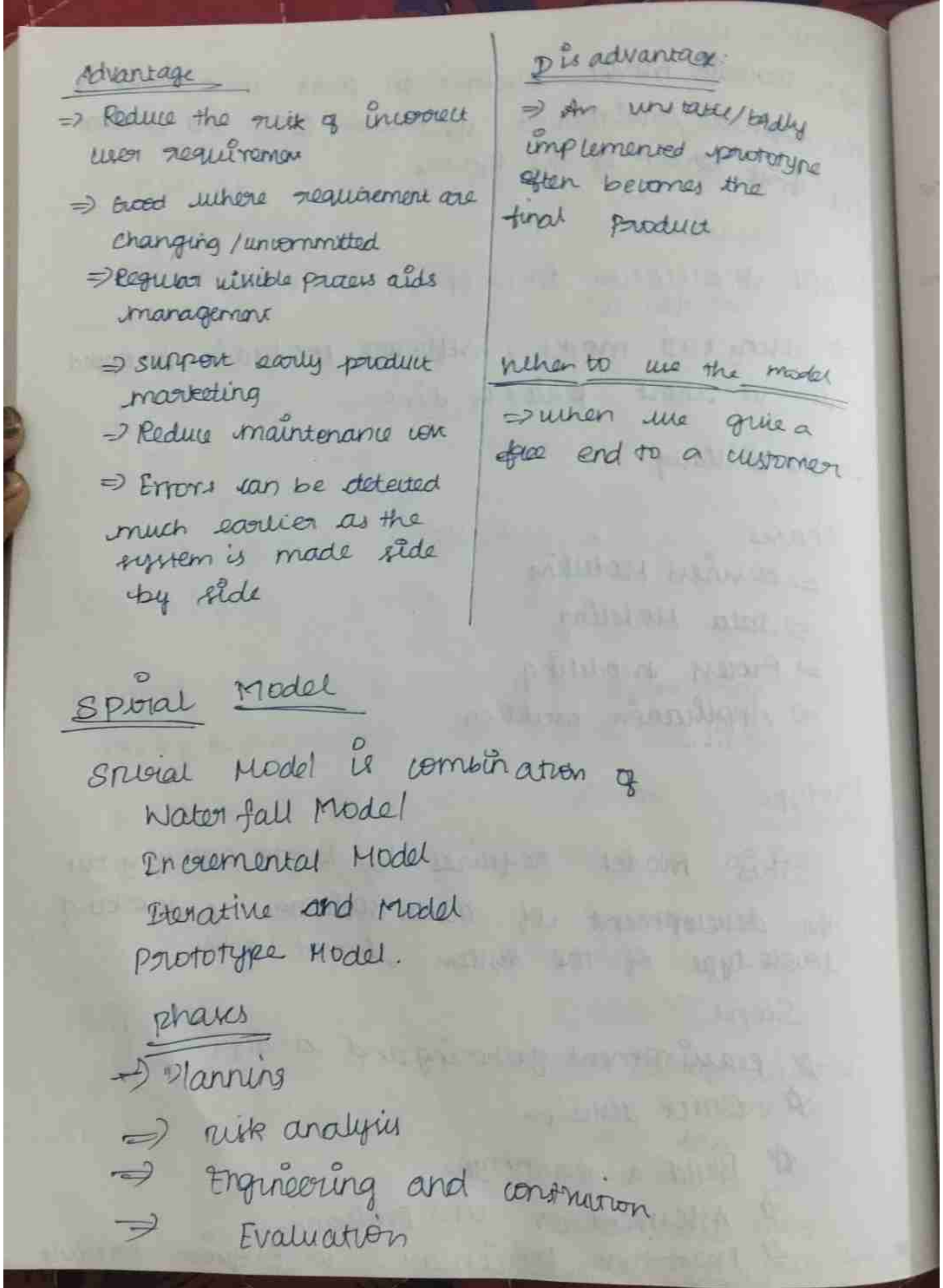
✰ Requirement gathering and analysis

✰ Quick decision

✰ Build a prototype

✰ Assessment or user Evaluation

✰ Prototype Retirement ✰ Engineer Product



Advantage

=> Reduce the risk of incorrect user requirement

=> Good where requirement are changing / uncommitted

=> Regular visible process aids management

=> support early product marketing

=> Reduce maintenance cost

=> Errors can be detected much earlier as the system is made side by side

Disadvantage

=> An unstable/badly implemented prototype often becomes the final product

When to use the model

=> when we give a free end to a customer

Spiral Model

Spiral Model is combination of
Water fall Model
Incremental Model
Iterative and Model
Prototype Model.

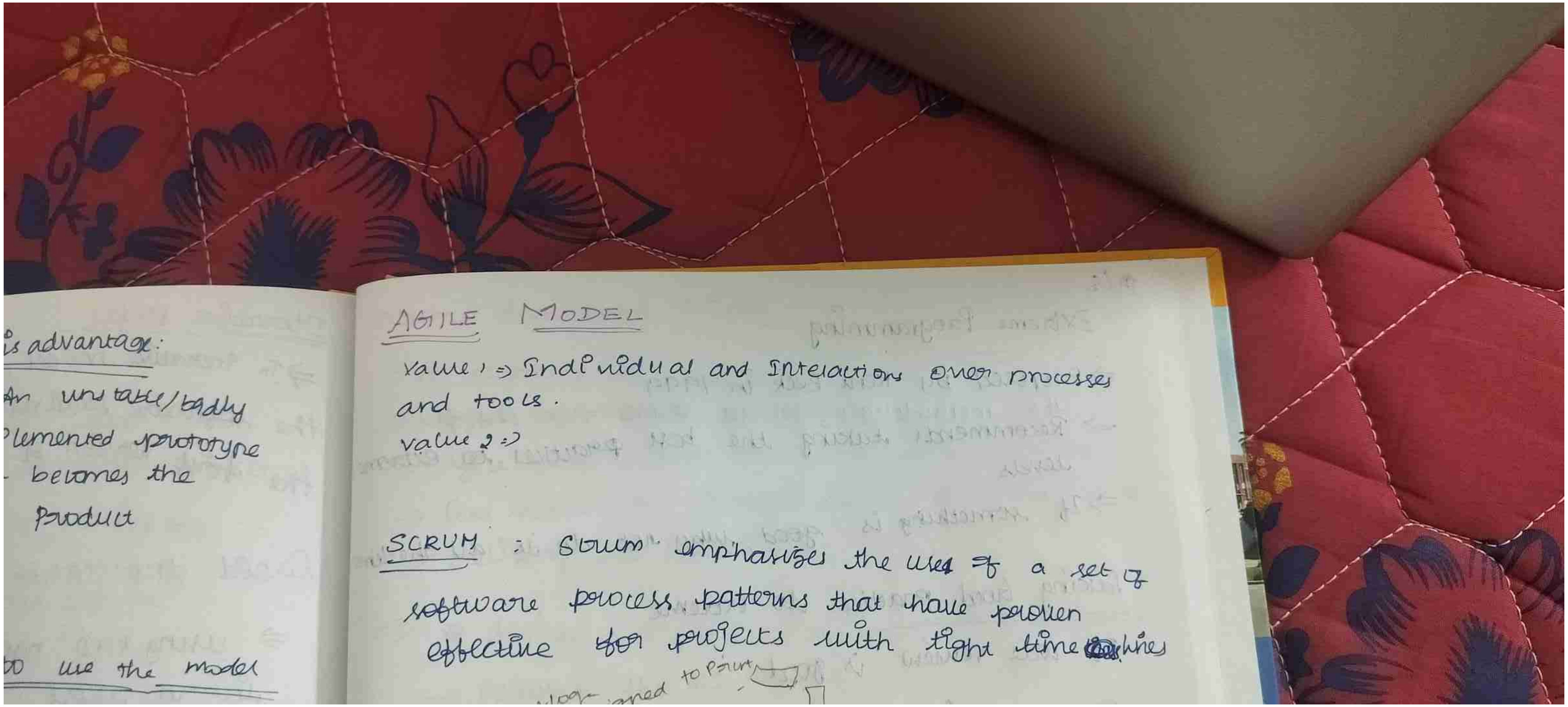
phases

=> Planning

=> risk analysis

=> Engineering and construction

=> Evaluation



is advantage:

An unstable/badly
plemented prototype
becomes the
Product

to use the model

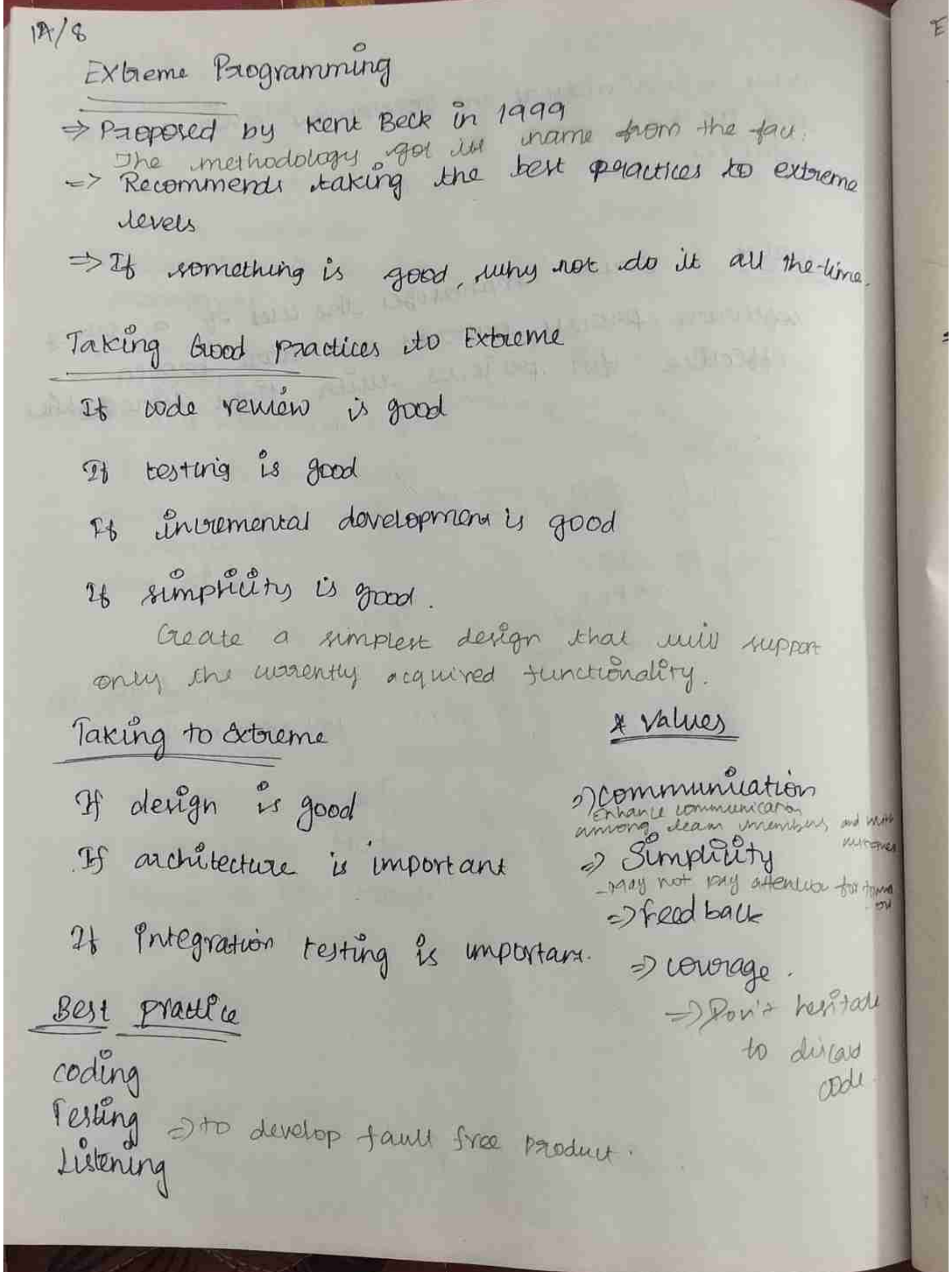
AGILE MODEL

value 1 ⇒ Individual and Interactions over processes and tools.

value 2 ⇒

SCRUM : Scrum emphasizes the use of a set of software process patterns that have proven effective for projects with tight timelines.

to Point



19/8

Extreme Programming

⇒ Proposed by Kent Beck in 1999

The methodology got its name from the fact:

⇒ Recommends taking the best practices to extreme levels

⇒ If something is good, why not do it all the time.

Taking Good practices to Extreme

If code review is good

If testing is good

If incremental development is good

If simplicity is good.

Create a simplest design that will support only the currently acquired functionality.

Taking to Extreme

If design is good

If architecture is important

If Integration testing is important.

Best practice

coding

Testing ⇒ to develop fault free product.

Listening

* Values

⇒ Communication
Enhance communication among team members and with customer

⇒ Simplicity
– may not pay attention for [illegible]

⇒ feed back

⇒ courage.
⇒ Don't hesitate to discard code

Extreme development Activity

⇒ XP planning an increment

⇒ begins by creating user stories

⇒ Agile team assesses each story and assigns a cost

→ few stories are grouped into a deliverable increment

⇒ delivery date planned

⇒ XP design

→ follows the KIS principle

→ Encourage the use of CRC cards

→ For difficult design problems, suggests the creation of "spike solution" — a design prototype

→ Encourages

Extreme program development activity

⇒ XP coding

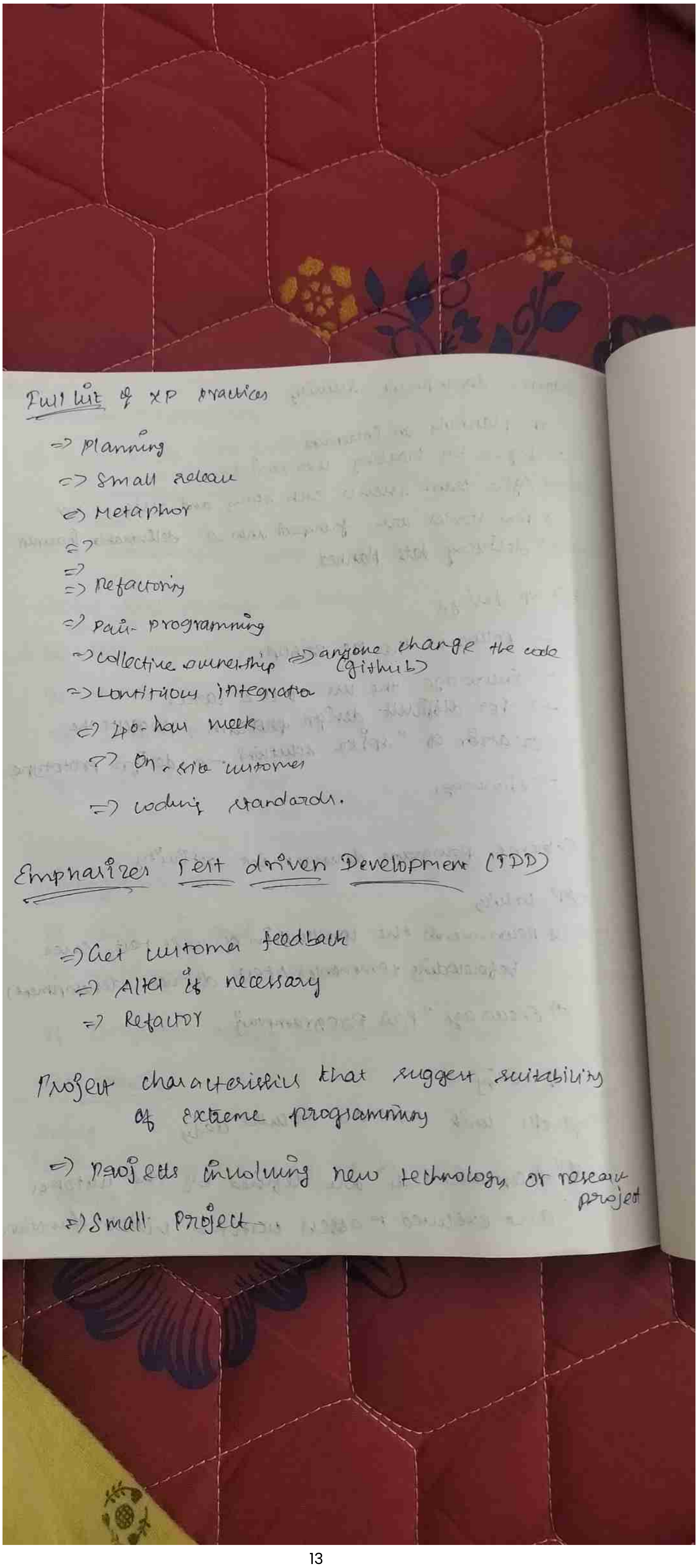
* Recommends the construction of unit test cases before coding commences (test-driven development)

* Encourage "Pair programming"

⇒ XP testing

* All unit tests are executed daily

* "Acceptance tests" are defined by the customer and executed to assess customer visible functionalities



Full list of XP practices

=> Planning
=> Small release
=> Metaphor
=>
=>
=> Refactoring
=> Pair programming
=> collective ownership => anyone change the code (github)
=> continuous integration
=> 40-hour week
=> On-site customer
=> coding standards.

Emphasizes Test driven Development (TDD)

=> Get customer feedback
=> Alter if necessary
=> Refactor

Project characteristics that suggest suitability of extreme programming

=> Projects involving new technology or research project
=> Small project