SOFTWARE PROTOTYPING

Yesoda Bhargava

“WHAT IS A SOFTWARE PROTOTYPE?

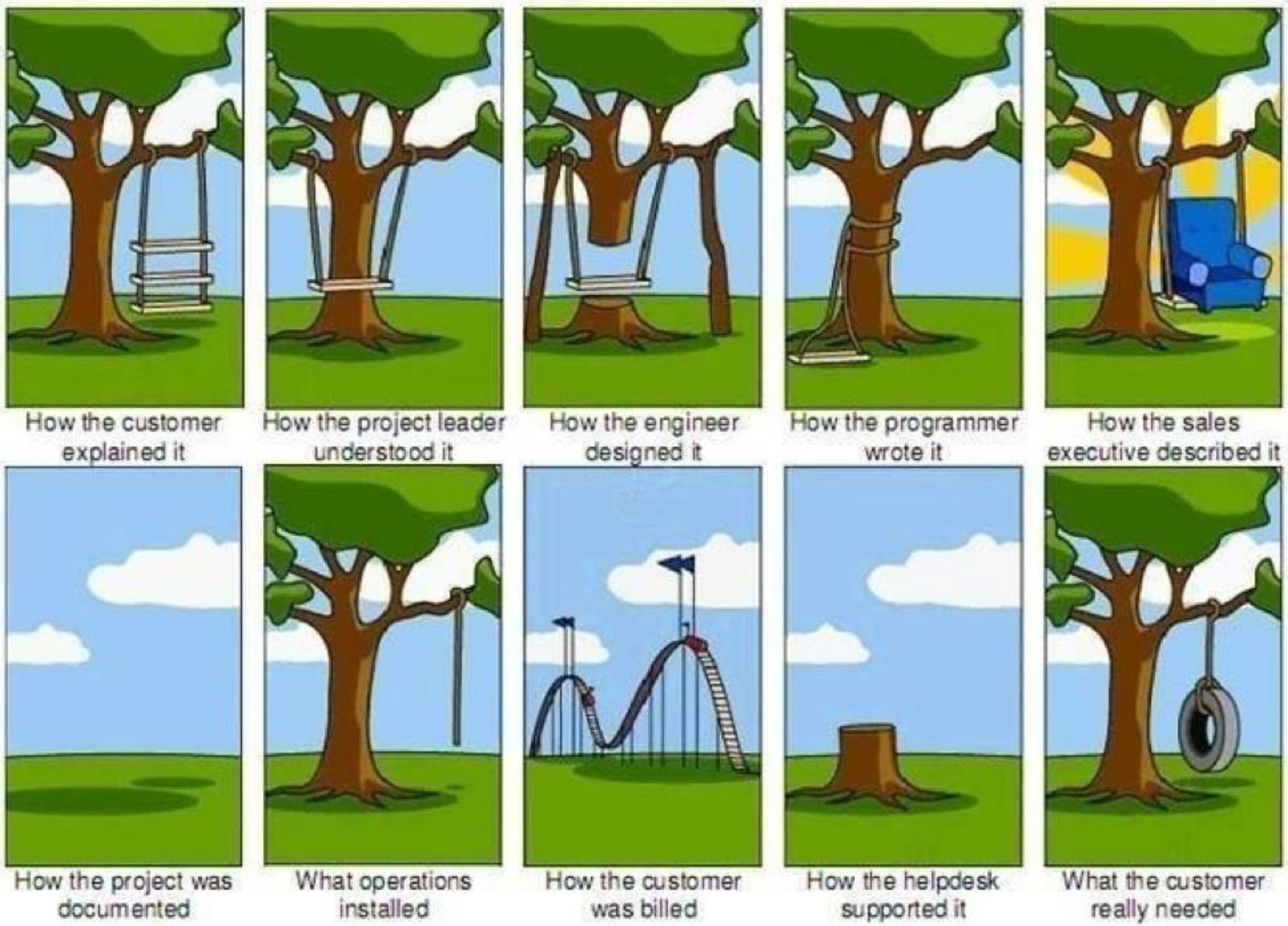
- Working model equivalent to a subset of the product.
- Like a toy implementation of a system having limited functional capabilities, low reliability and inefficient performance.
- Model generally illustrated to customers along with input data formats, messages, and interactive dialogs.
- Technical issues associated with the product are critically examined.
- **It is not a production code, but it may become pre-production code or may be discarded entirely.**

WHAT'S THE IMPORTANCE OF CREATING A SOFTWARE PROTOTYPE?

- In which model did you hear first the word Prototype?
- Prototyping is the technology which laid the foundation for the Spiral Model. The founder's experiments actually showed that prototyping reduces programmer's effort by 40%.
- Software development theory focuses on static behaviour of the software. Little theory on its dynamic behaviour : how it performs under load.
- **Software have the propensity to fail without warning. Reducing the risk of the product failure is the key goal when it comes to prototyping.**

THE FUNDAMENTAL IMPORTANCE OF SOFTWARE PROTOTYPE

- Provides a vehicle for systems engineers to better understand the environment and the requirements problem being addressed.
- An overview of what is feasible with the existing technology and where the technical weak spots exist.
- Promising mechanism for the transfer of design intent from system engineer to the developer.
- Allows early customer interaction. This means earlier feedback of the running of the software.
- Allows evolution of system requirements.
- Software prototyping refers to the process of visualising a software product before it has been created.



How the customer explained it

How the project leader understood it

How the engineer designed it

How the programmer wrote it

How the sales executive described it

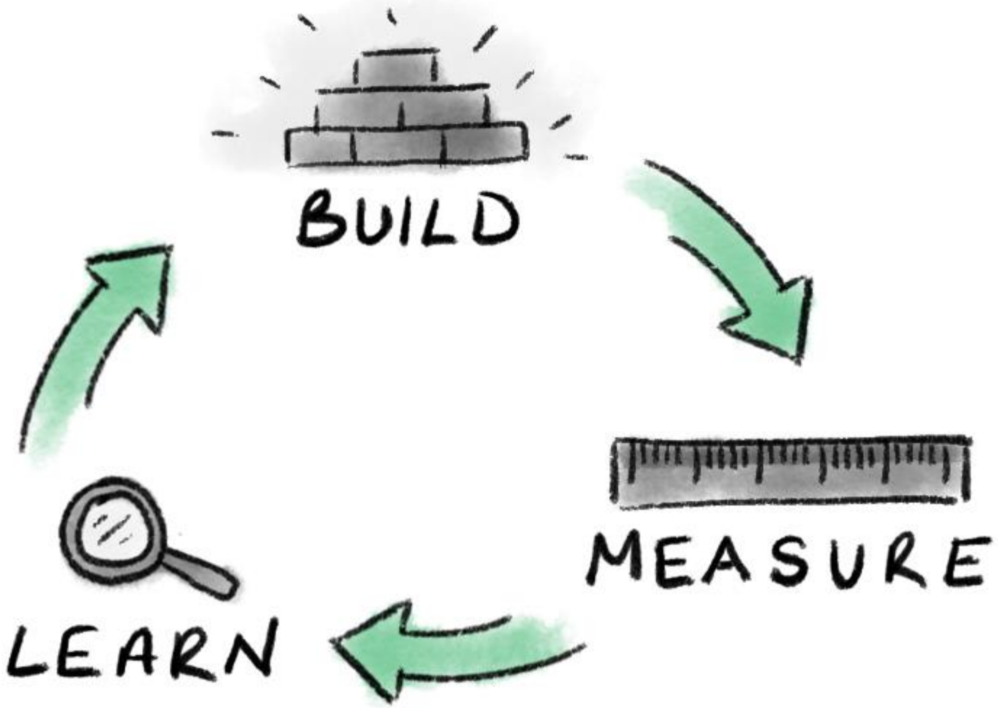
How the project was documented

What operations installed

How the customer was billed

How the helpdesk supported it

What the customer really needed


BUILD
MEASURE
LEARN

POC VS PROTOTYPE VS MVP

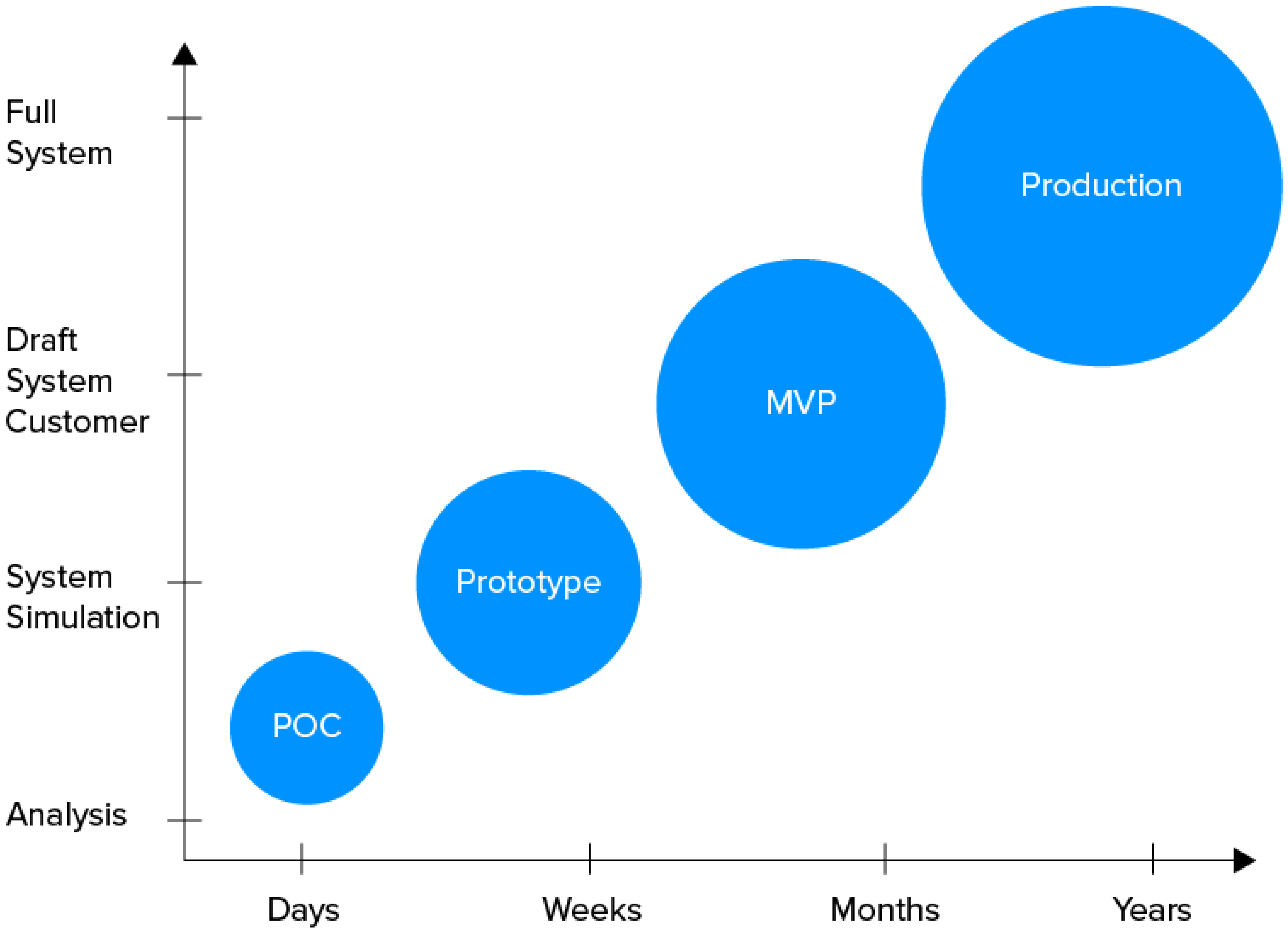
- Proof of Concept is usually a short-term project aimed at proving the validity of applying a given business idea in a specific company.
- The PoC must prove the feasibility of the idea and the possibility of achieving the assumed business results.
- It is used before the actual product development and before the product is brought to market.
- When developing a POC, a small project should be implemented to test specific business concepts and technical capabilities.
- Developing a POC is the fastest and most accurate way to test or invalidate assumptions about target users and application concepts.

- A prototype is a sample, preliminary visualization, or product released to test the concept.
- The prototype strategy is usually used to detect errors in the system.
- While creating a prototype, developers test product design, usability, and functionality.
- Prototypes help create the appearance and logic of the application.
- It helps test how customers use and react to a given project.
- Like a trial version of the product.

- Minimum Viable Product is a product in early development, with a minimum set of kits that is sufficient to launch it on the market for customers seeking an initial evaluation.
- The whole concept is based on testing your business ideas against real, hard numbers generated by real customers, rather than on hypothetical values.

MVP EXAMPLE: TWITTER

- The originators created key features for the idea and tested them on a limited group of users.
- In 2006, the Odeo company created a prototype of a social network that allows employees to send messages called “Twitts” of max. 160 characters long.
- After successful tests, the website was released to the market in the same year.
- Known as a social networking site for microblogging, with time it turns out to be primarily a platform enabling fast and concise information exchange.

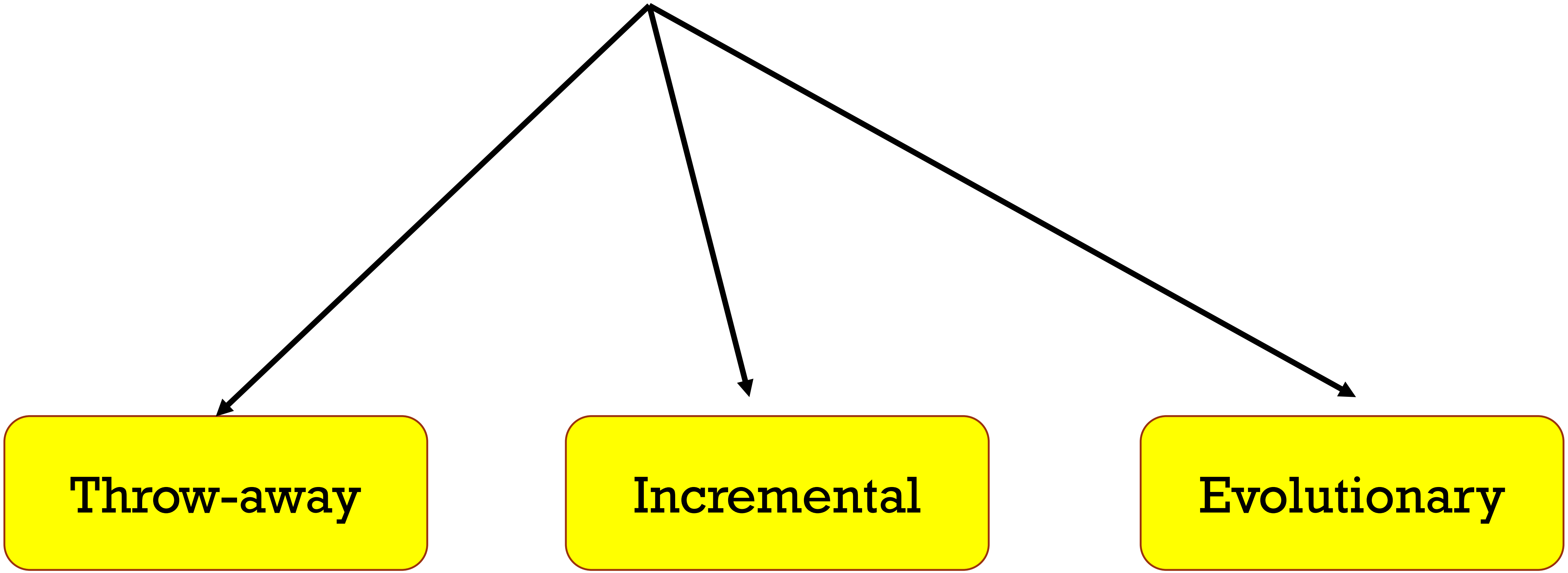

Full System
Draft System Customer
System Simulation
Analysis
POC
Prototype
MVP
Production
Days
Weeks
Months
Years

Don’t write the strategy down, make a prototype.

Doesn't construction of a working model involve additional costs???

But the overall development cost is lower.

CLASSIFICATION OF SOFTWARE PROTOTYPES



Throw-away

Incremental

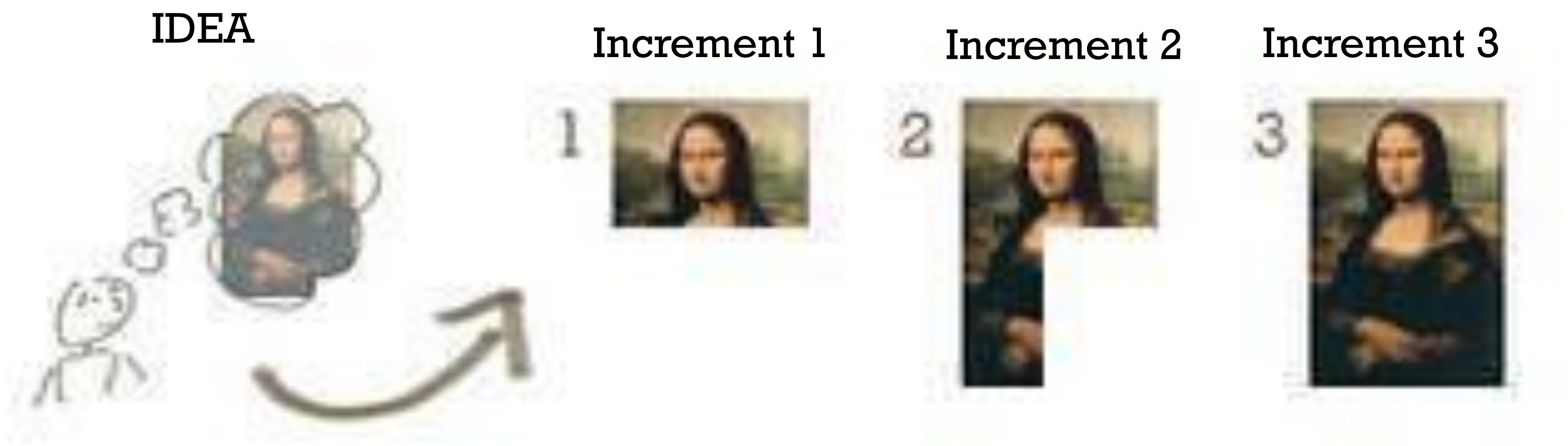
Evolutionary

THROW AWAY PROTOTYPES

- Used only to test out some ideas.
- Discarded later when the development of an operation system is commenced.
- Such prototype could be developed using a different software environment where the machine efficiency is important or even on a different hardware platform.
- Also called as "close ended prototypes".
- Cheap, fast, designed to model an idea or feature.
- It can enable early customer engagement.
- Short project timeline and easier and faster to develop the interface.
- Not much in use now.

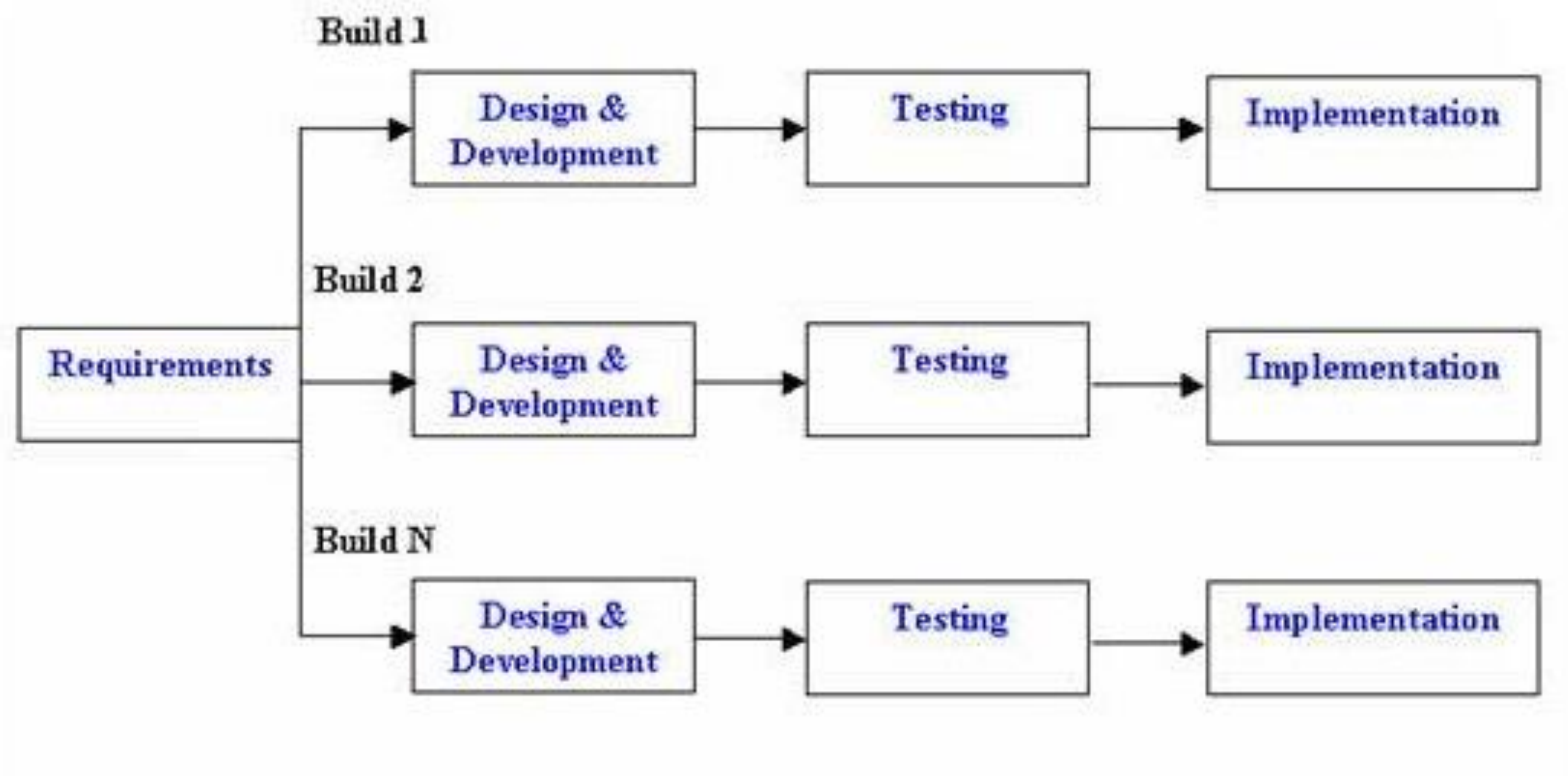
INCREMENTAL PROTOTYPE

- Breaking the system down into small components that are then implemented and delivered in sequence.
- Feedback from early increments can influence later stages.
- Smaller sub-projects are easier to control and manage.
- However, later increments may require the earlier increments to be modified due to software breakage.
- Suitable when? Major functional requirements of the system are understood.
- Useful for enterprise software that has many modules and components.



General idea of the incremental design.

1. Addition of piece in each increment, but expecting that each piece is fully finished.
2. Keep adding the pieces until it is complete.



Incremental Prototype Model.

Advantages

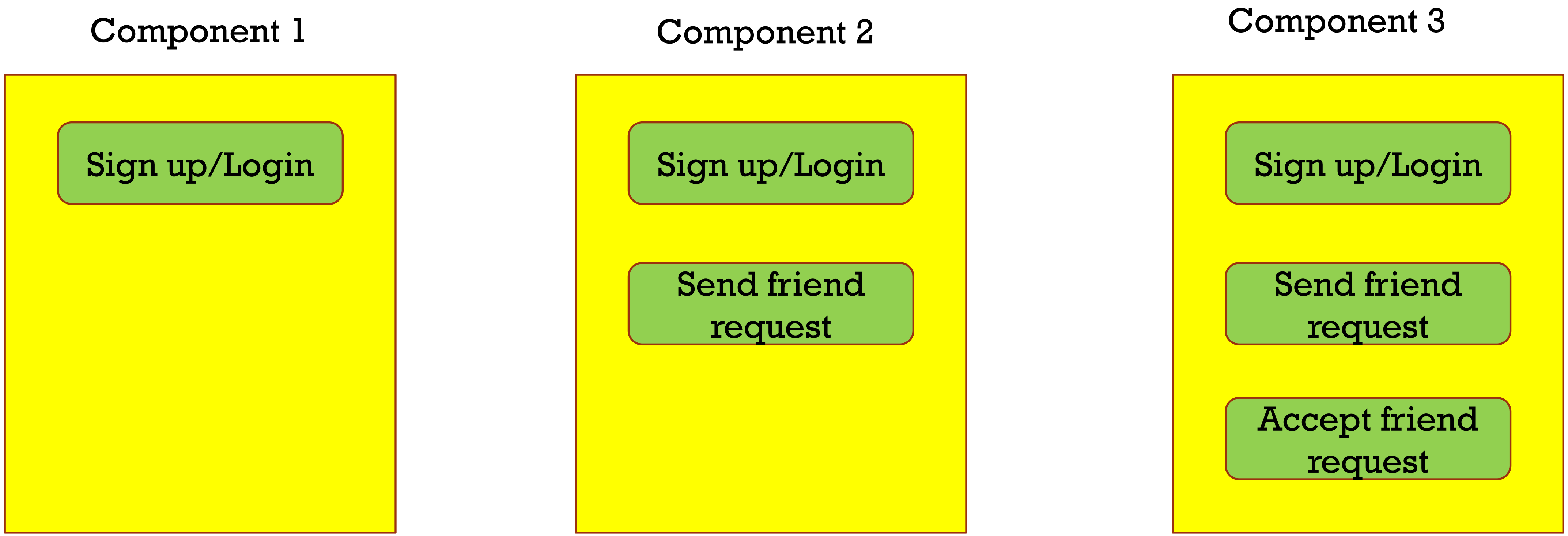
- Generates working software quickly.
- Model Is flexible – less costly to change scope and requirements.
- Easier to test and debug during a smaller iteration.
- Risk management and assessment is easier.
- Reduces the complexity of the development process.

Disadvantages

- Demands clear planning and design.
- A clear and complete definition of the whole system before it can be broken down and built incrementally.

CASE STUDY OF INCREMENTAL PROTOTYPE

- Example : you want to develop a web-based social network with following functional requirements:
  - Sign up
  - Log in and send/accept friend request.



Component 1

Sign up/Login

Component 2

Sign up/Login

Send friend request

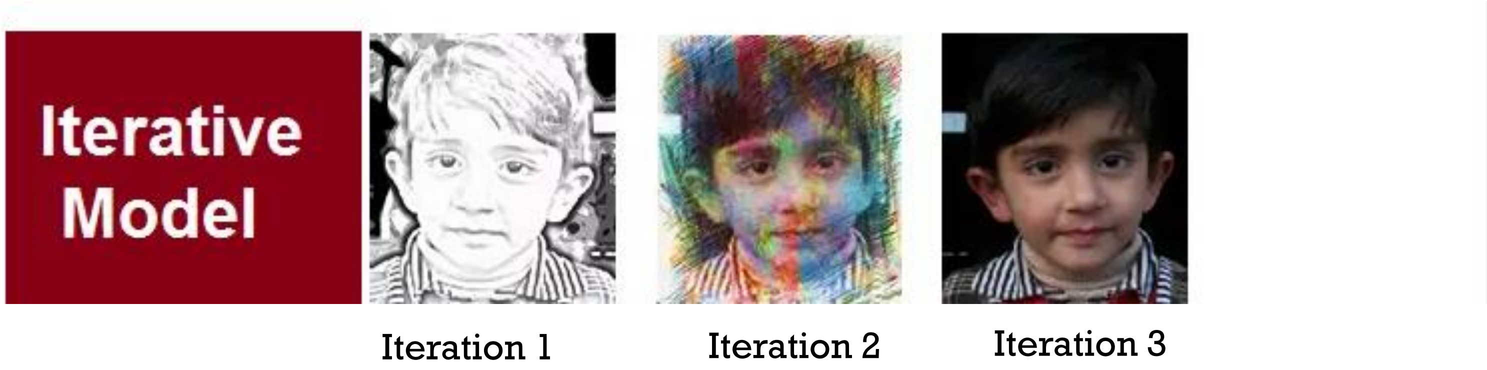
Component 3

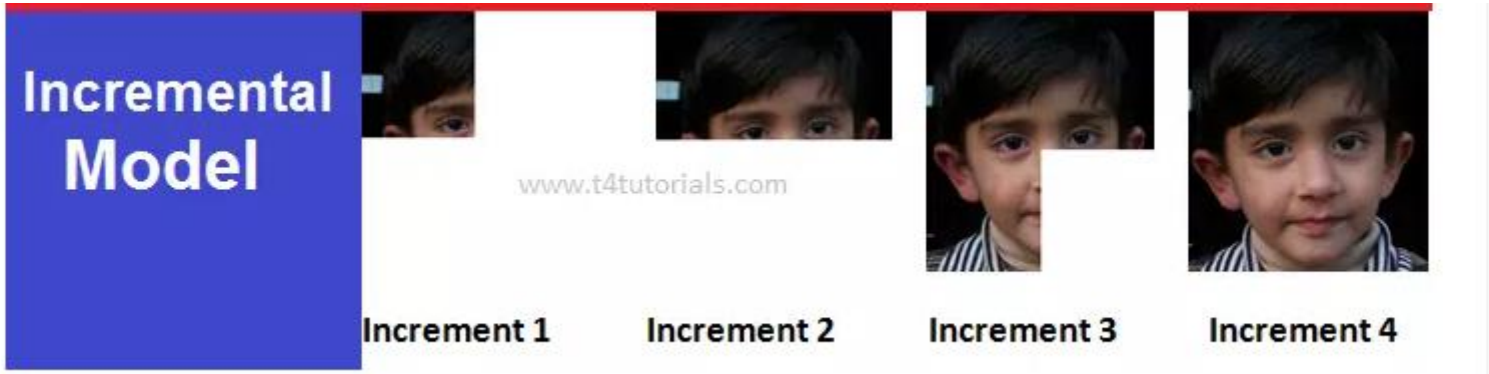
Sign up/Login

Send friend request

Accept friend request

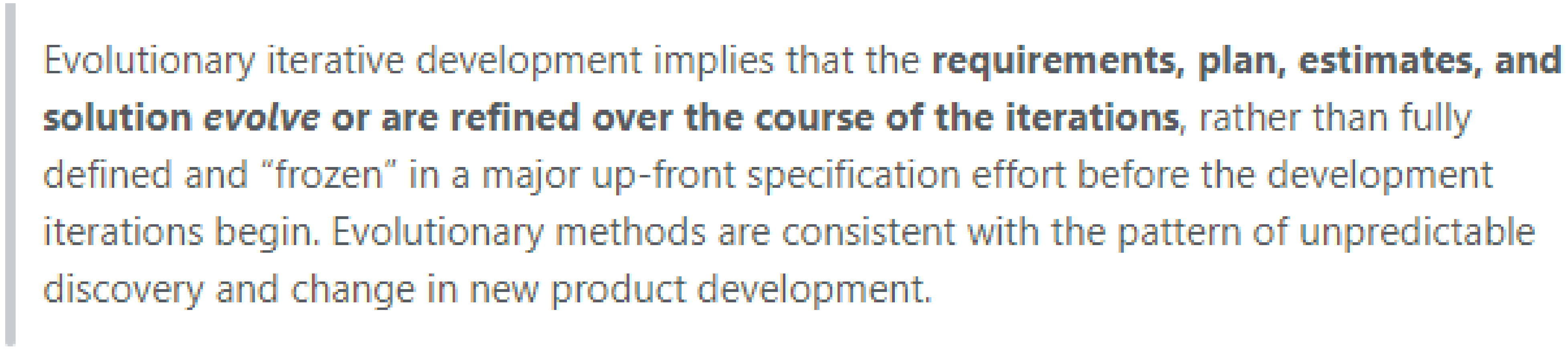
WHAT'S THE DIFFERENCE BETWEEN INCREMENTAL AND ITERATIVE MODEL?





EVOLUTIONARY PROTOTYPING

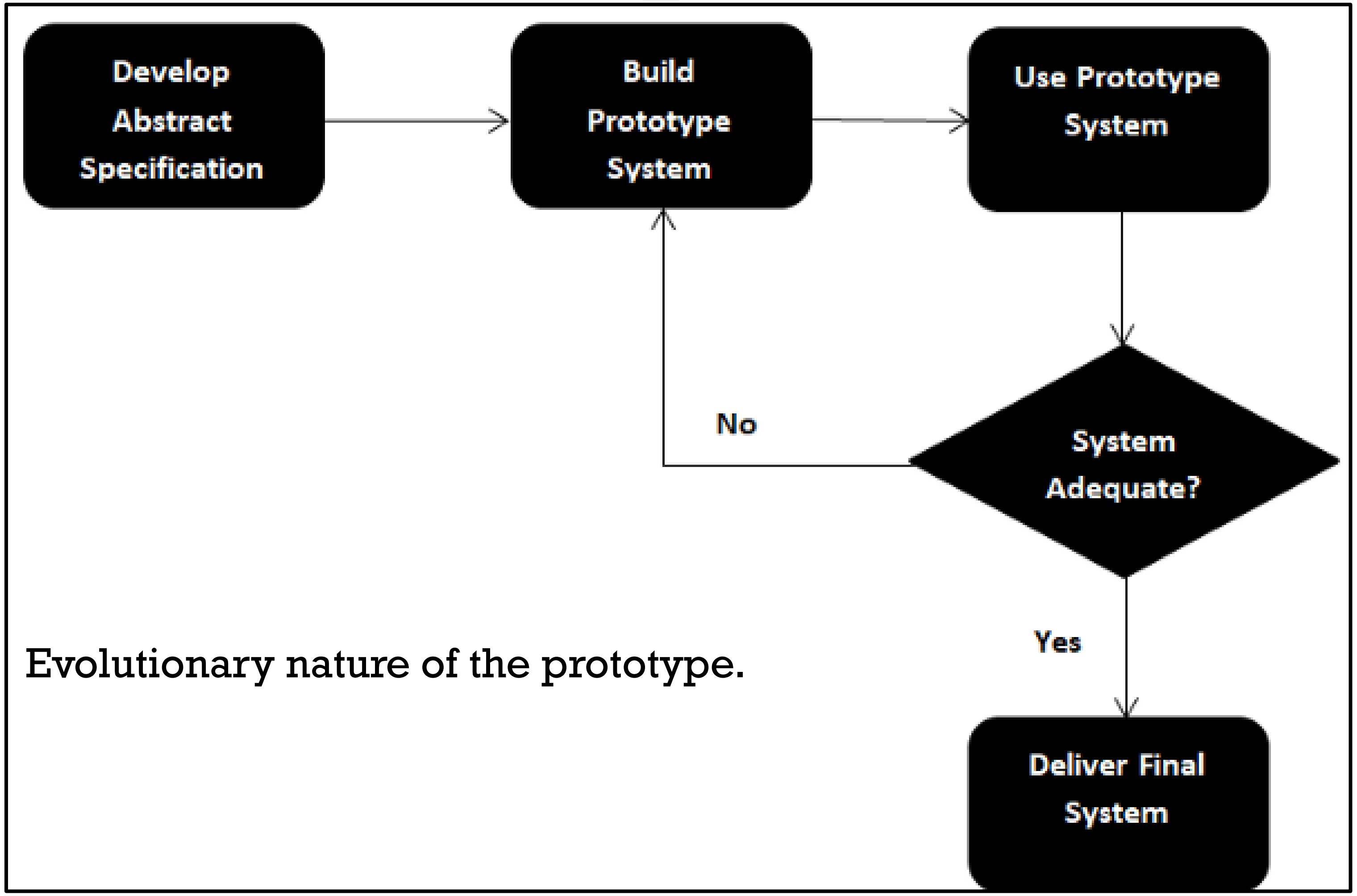
- Most fundamental form of prototyping, also known as BREADBOARD PROTOTYPING.
- Main Concept: build a robust prototype and constantly improve it.
- Allows developers to add features or make changes that couldn't be devised during the requirements analysing and designing.
- It assists in new ways to improve the system.
- Incrementally refined on the basis of customer feedback till it finally gets accepted.
- Drawbacks:
  - Expensive long term maintenance.



> Evolutionary iterative development implies that the **requirements, plan, estimates, and solution *evolve* or are refined over the course of the iterations**, rather than fully defined and “frozen” in a major up-front specification effort before the development iterations begin. Evolutionary methods are consistent with the pattern of unpredictable discovery and change in new product development.

Functional piece of software

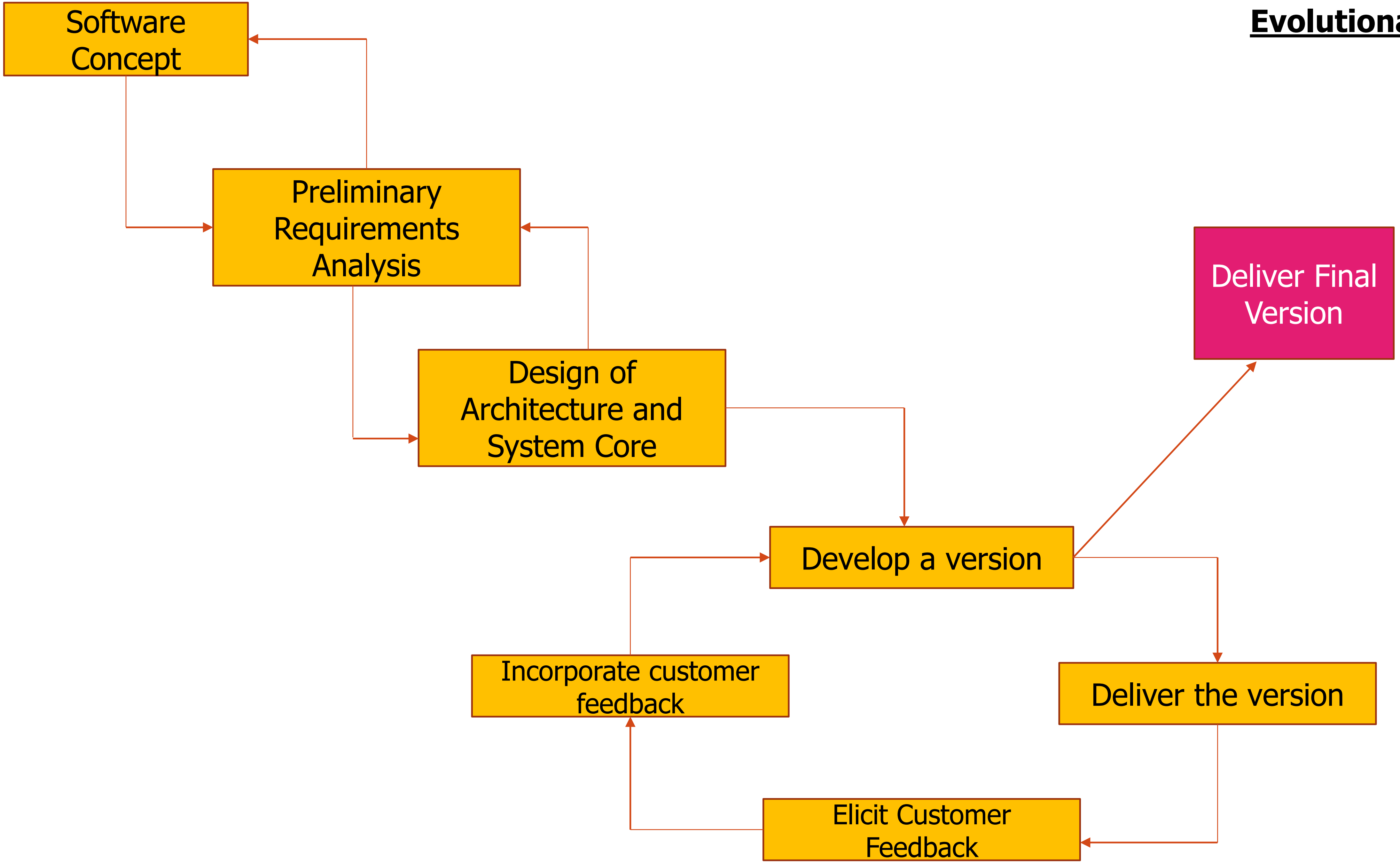
It would not do everything the customer requires, but it makes a good starting point.

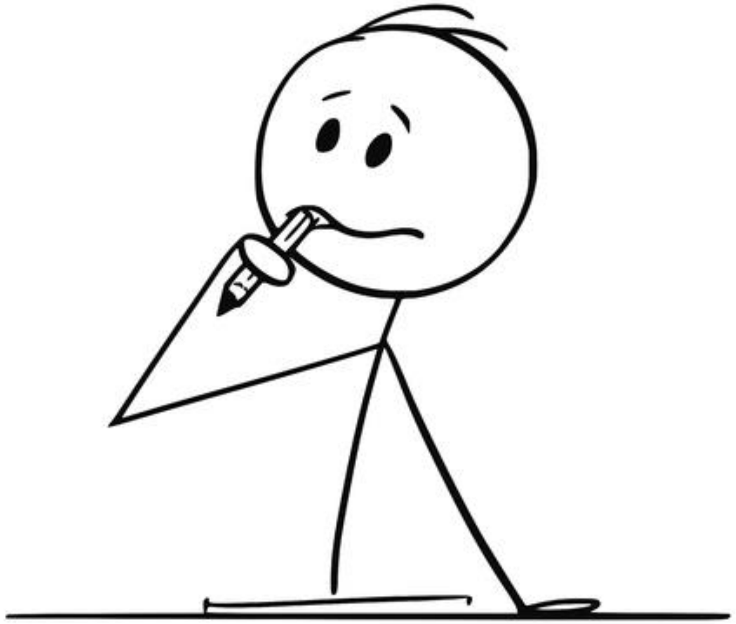


Evolutionary nature of the prototype.

The first iteration of an evolutionary prototype is similar to the minimum viable product (MVP)

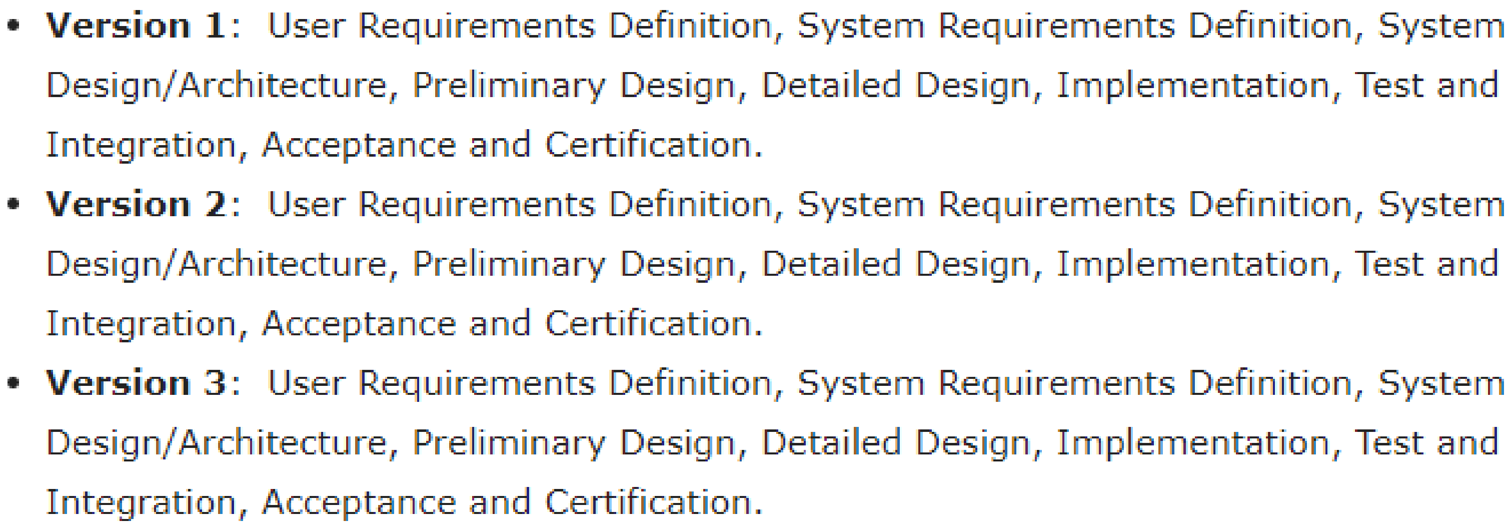
Evolutionary Delivery


Software Concept
Preliminary Requirements Analysis
Design of Architecture and System Core
Develop a version
Deliver Final Version
Deliver the version
Elicit Customer Feedback
Incorporate customer feedback



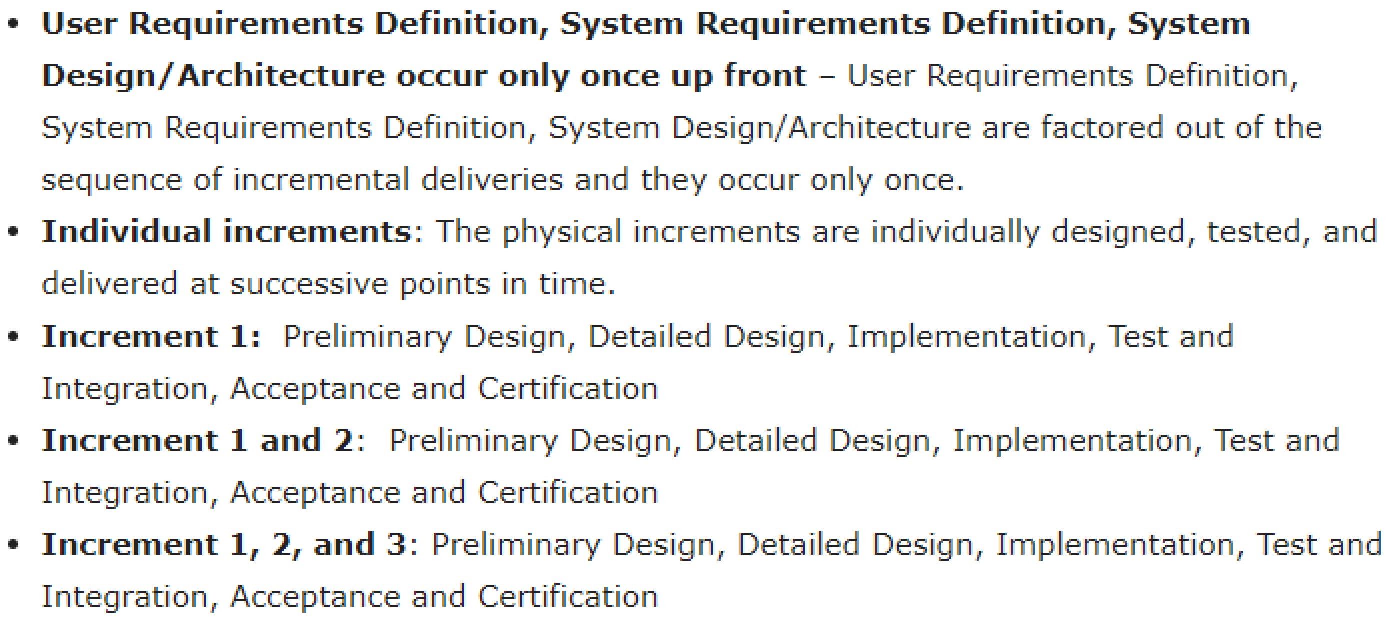
I am confused between incremental and evolutionary prototype model.

EVOLUTIONARY PROTOTYPE



- **Version 1**: User Requirements Definition, System Requirements Definition, System Design/Architecture, Preliminary Design, Detailed Design, Implementation, Test and Integration, Acceptance and Certification.
- **Version 2**: User Requirements Definition, System Requirements Definition, System Design/Architecture, Preliminary Design, Detailed Design, Implementation, Test and Integration, Acceptance and Certification.
- **Version 3**: User Requirements Definition, System Requirements Definition, System Design/Architecture, Preliminary Design, Detailed Design, Implementation, Test and Integration, Acceptance and Certification.

INCREMENTAL PROTOTYPE



- **User Requirements Definition, System Requirements Definition, System Design/Architecture occur only once up front** – User Requirements Definition, System Requirements Definition, System Design/Architecture are factored out of the sequence of incremental deliveries and they occur only once.
- **Individual increments**: The physical increments are individually designed, tested, and delivered at successive points in time.
- **Increment 1:** Preliminary Design, Detailed Design, Implementation, Test and Integration, Acceptance and Certification
- **Increment 1 and 2**: Preliminary Design, Detailed Design, Implementation, Test and Integration, Acceptance and Certification
- **Increment 1, 2, and 3**: Preliminary Design, Detailed Design, Implementation, Test and Integration, Acceptance and Certification

EXTREME PROTOTYPE

- Method mainly used for web development.
- Consists of three sequential independent phases:
  - In this phase a basic prototype with all the existing static pages are presented in the HTML format.
  - In the 2nd phase, functional screens are made with a simulate data process using a prototype services layer.
  - This is the final step where all the services are implemented and associated with the final prototype.
- With this approach, the user interface is designed and developed before any of the underlying technology is implemented, which is what gives it the “extreme” tag.

DIS-ADVANTAGES OF PROTOTYPING

- Slow and time-taking process.
- Far too many variations in the software when each prototype is evaluated by the customer.
- Difficult to maintain a proper documentation because the requirements of the customers are changing.
- Danger of losing client if they are not satisfied with the prototype.
- Developers may not always understand the stake in building prototype and may create sub-standard solutions.

READING RECOMMENDATIONS

- Use of AI in Software Development