SOFTWARE ENGINEERING

Monika Gupta

IBM Research

SOFTWARE TESTING

Testing: Definitions

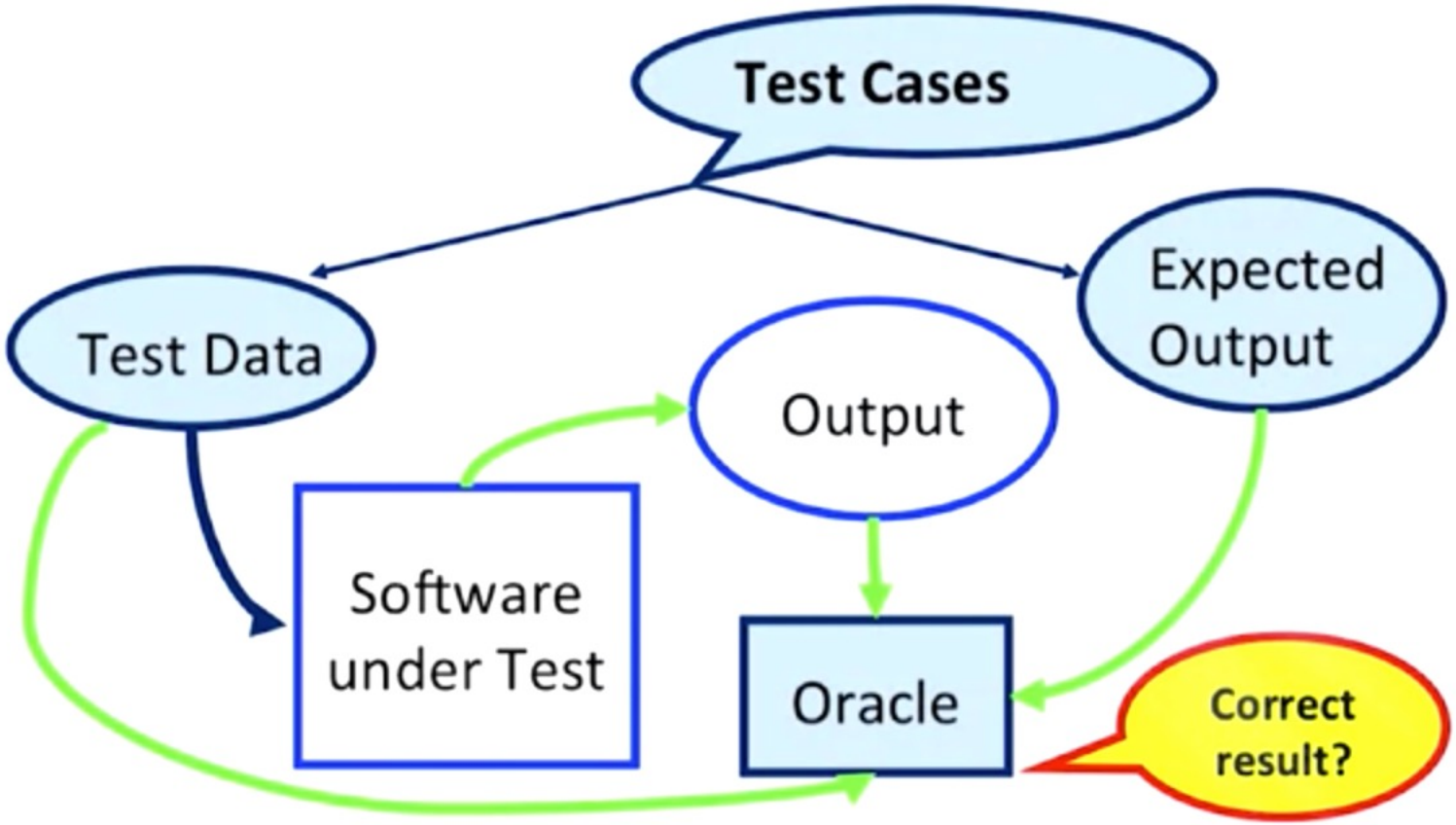
The process of executing a program (or part of a program) with the intention of finding errors(Myers, via Humphrey)

The purpose of testing is to find errors

Testing is the process of trying to discover every conceivable fault or weakness in a work product (Myers, via Kit)

The process of searching for errors (Kaner)

What is a Test?



Test data and test cases

Test Data

Inputs which have been devised to test a system

Test Cases

Inputs to the system and the predicted outputs from operating the system on these inputs, if the system performs to its specification

What is a “bug”?

Failure

Error

- Latent
- Effective

Fault

“Dependable computing and fault tolerance: concepts and terminology” - Jean-Claude Laprie, 1985.

So, the error is a manifestation of a fault.
And a failure is the manifestation of an error on the servers.

Axiom of Testing

Program testing can be used to show the presence of bugs, but never their absence.

-Dijkstra, 1969

Software testing is a process to find (and fix) defects in a system's implementation

There is no way to prove a program correct through testing

Still an essential part of the process

Must ensure quality of input selection and correct expected output

Understanding testing is key to a successful project

Verification & Validation: IEEE

Verification

> The process of evaluating a system or component to determine whether the products. . . satisfy the conditions imposed. . .

Validation

> The process of evaluating a system or component. . . to determine whether it satisfies specified requirements.

Verification & Validation: Kaner

Verification

Checking a program against the most closely related design documents or specifications

Validation

Checking the program against the published user or system requirements

Verification & Validation: Myers

Verification

An attempt to find errors by executing a program in a test or simulated environment

Validation

An attempt to find errors by executing a program in a real environment

VERIFICATION AND VALIDATION

Verification

- Confirm that the software performs and conforms to its specification
- "Are we building the thing right?"

Validation

- Confirm that the software performs to the user's satisfaction
- Assure that the software system meets the user's needs
- "Are we building the right thing?"

Dynamic vs. Static Verification

Dynamic V & V

- Concerned with exercising and observing product behavior
- Testing (in all forms)

Static V & V

- Concerned with analysis of the static system representations to discover problems
- Proofs and inspections

Verification and Validation

Building the thing right and building the right thing

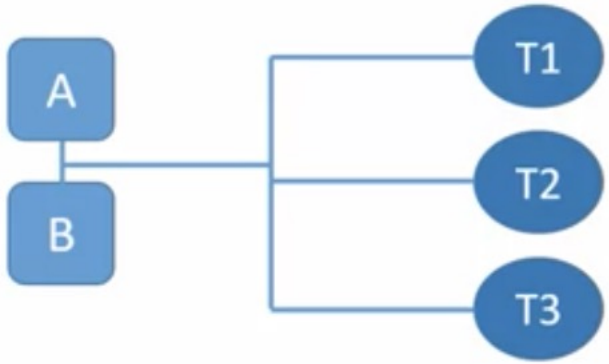
Regardless of definition, an important aspect of testing

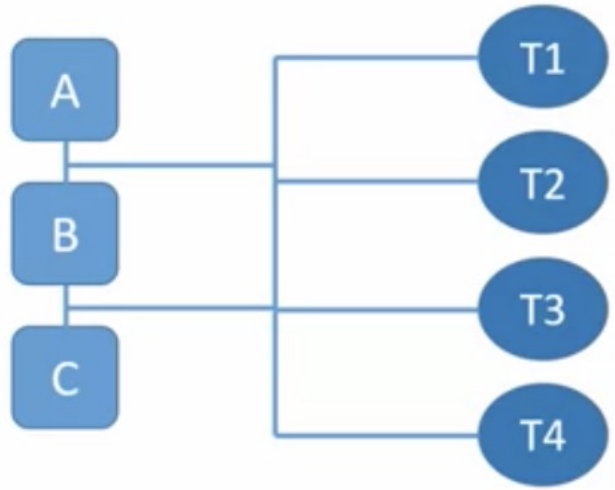
Dynamic and static V&V aren't exclusive

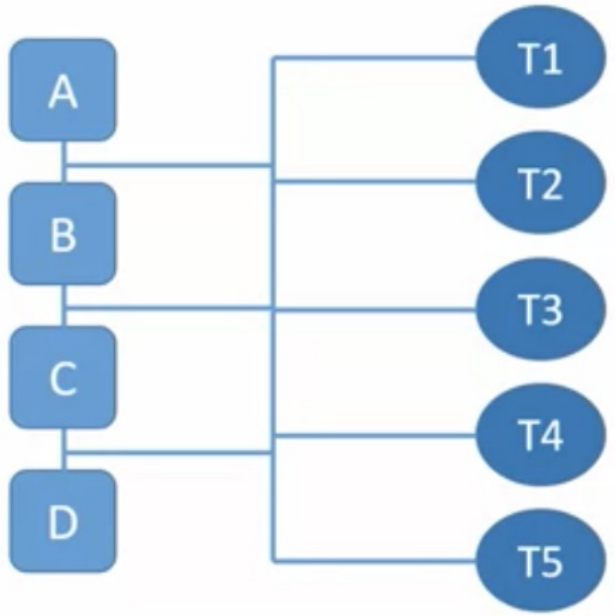
Verification is cheaper/easier than Validation

SOFTWARE TESTING STRATEGIES

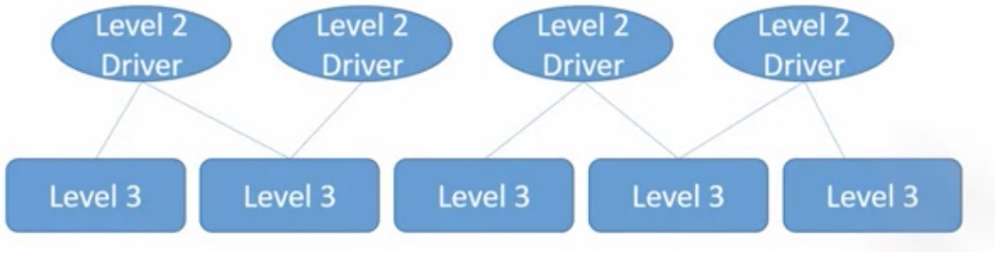
Incremental Testing

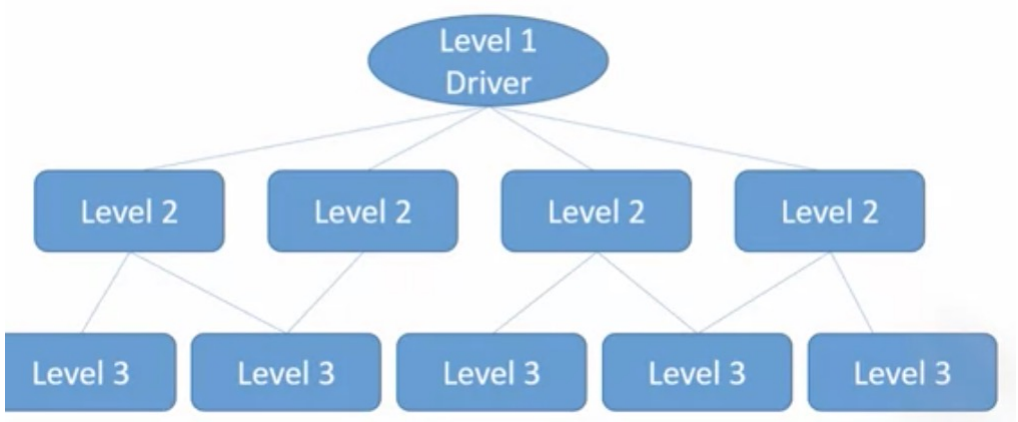




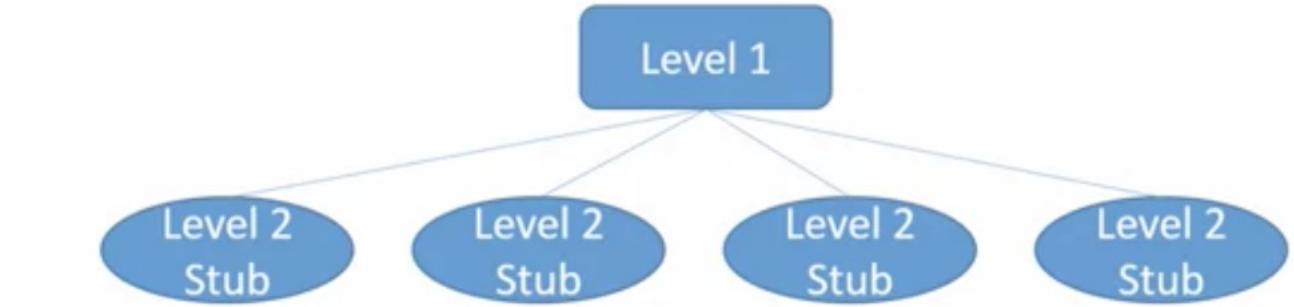


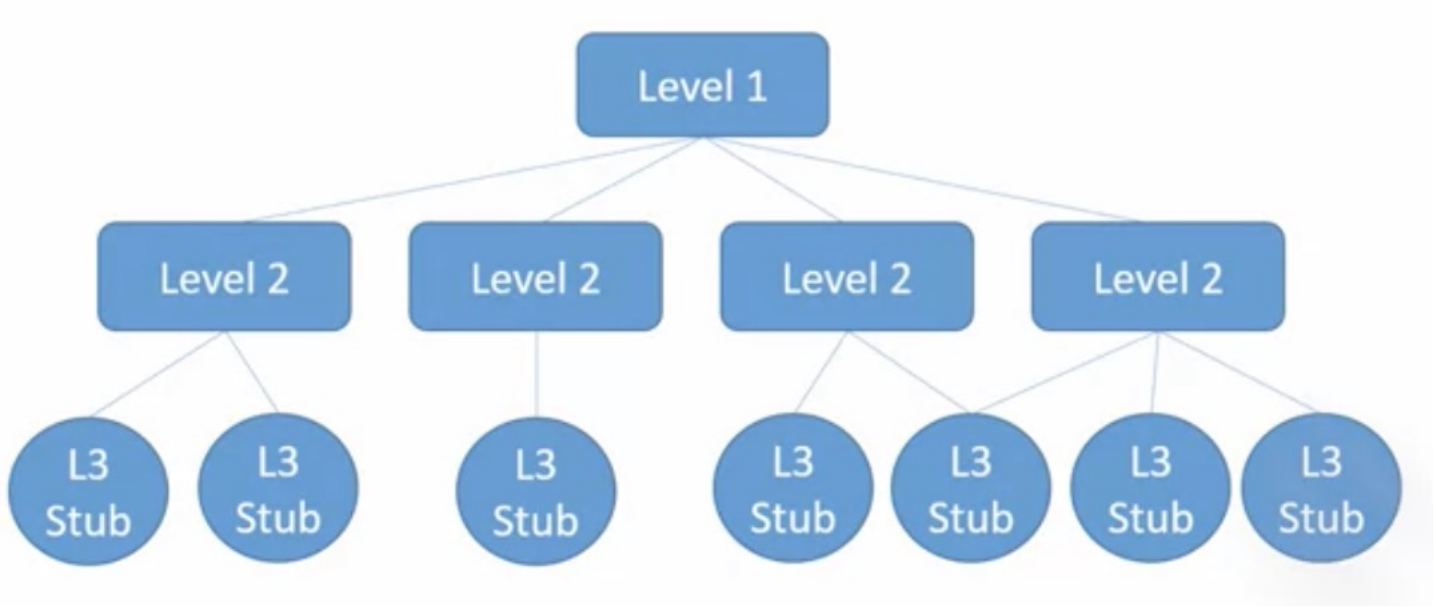
Bottom-up Testing

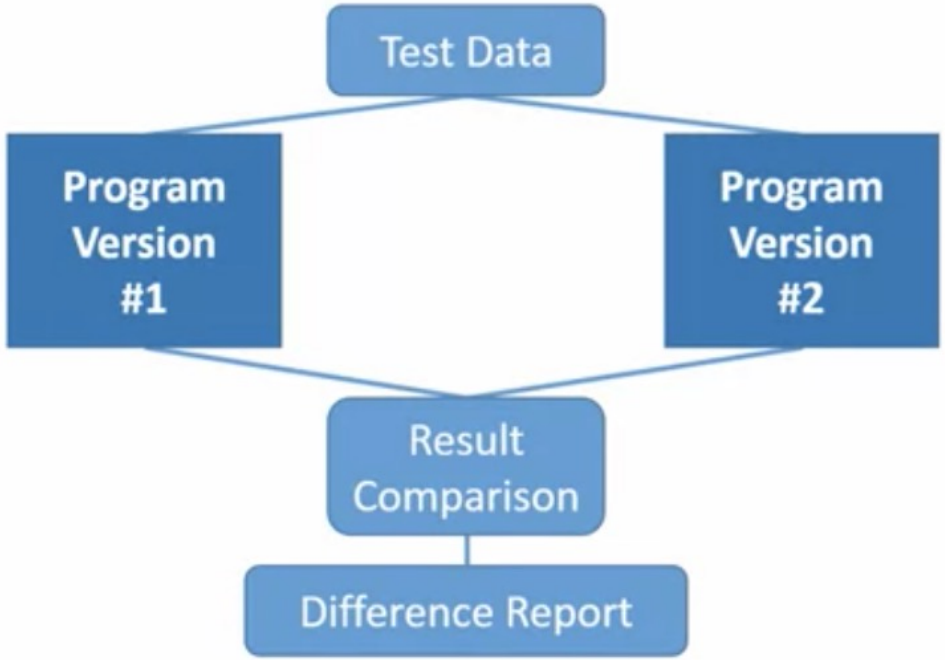




Top-down Testing







BACK TO BACK TESTING

Test Scaffolding

GOAL: Setup the environment for executing tests

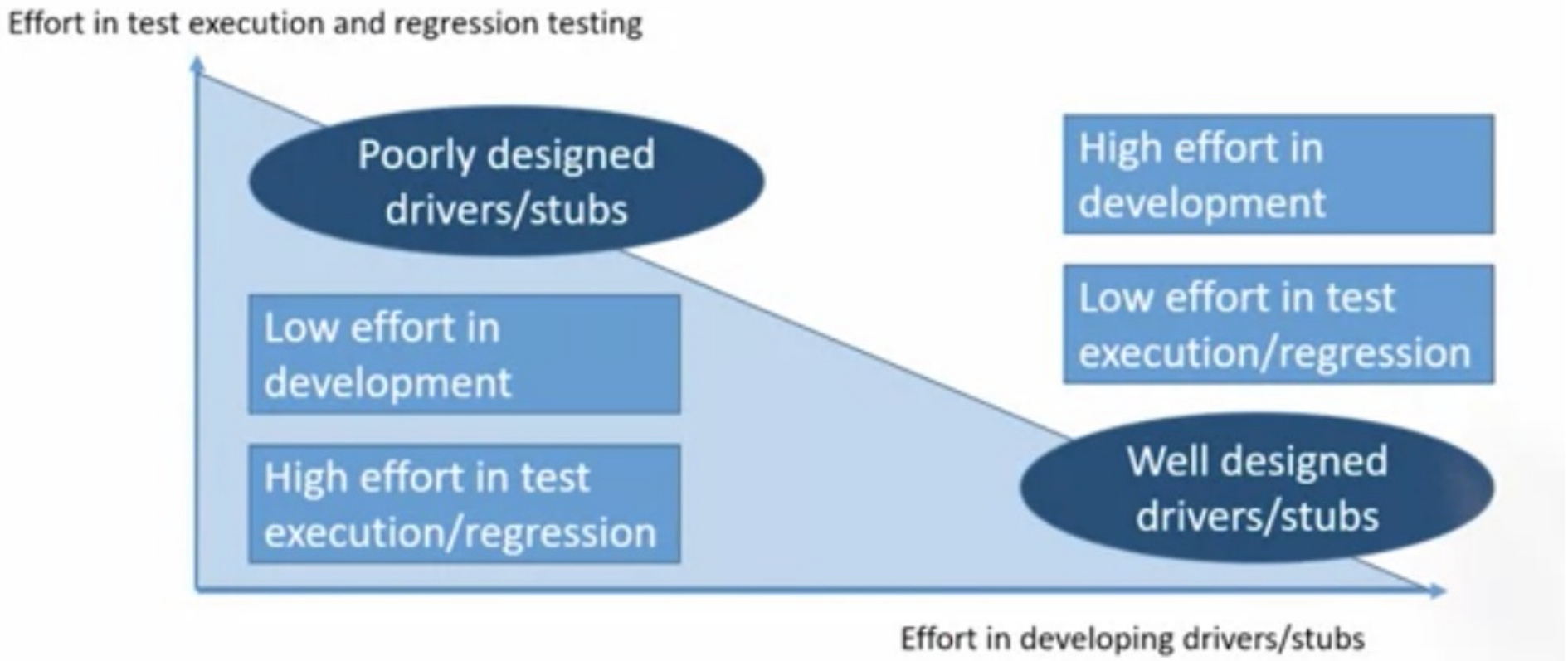
Driver

Stub

Program Unit

Oracle

Tradeoff


Effort in test execution and regression testing
Poorly designed drivers/stubs
Low effort in development
High effort in test execution/regression
High effort in development
Low effort in test execution/regression
Well designed drivers/stubs
Effort in developing drivers/stubs

Who should test?

Developer

- "This is my baby!"
- Understands the system
- Will test gently
- Driven by deadlines

Tester

- "Smashy smashy!"
- Must learn the system
- Will attempt to break it
- Driven by "quality"

AXIOMS OF TESTING

As the number of detected defects in a piece of software increases, the probability of the existence of more undetected defects also increases

Assign your best programmers to testing

Exhaustive testing is impossible

You cannot test a program completely

Even if you do find the last bug, you'll never know it

It takes more time than you have to test less than you'd like

You will run out of time before you run out of test cases

Strategies of testing drive the actual act of testing units

Pure top-down or bottom-up doesn't exist

Stubs are created to allow higher level units to execute

Drivers organize and execute the units under test

Programs cannot be tested completely

SOFTWARE TESTING PERSPECTIVES

Black-box and White-box Testing

Black-box Testing

- Designed without knowledge of program's internal structure
- Based on functional requirements

White-box Testing

- Examines the internal design of the program
- Requires detailed knowledge of its structure

V & V Process

Must be applied at each stage of the process

Has two principal objectives

- Discovery of defects in a system
- Assessment of whether or not the system is usable

Stages of Testing

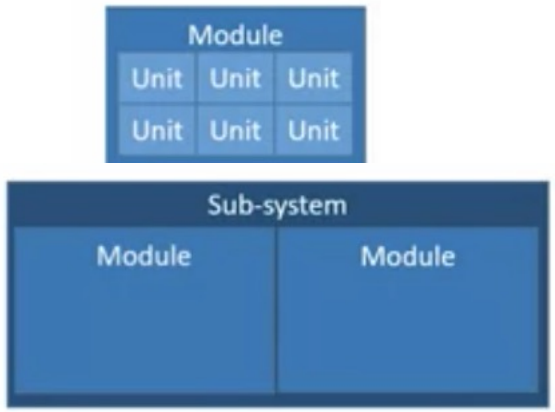
Unit testing

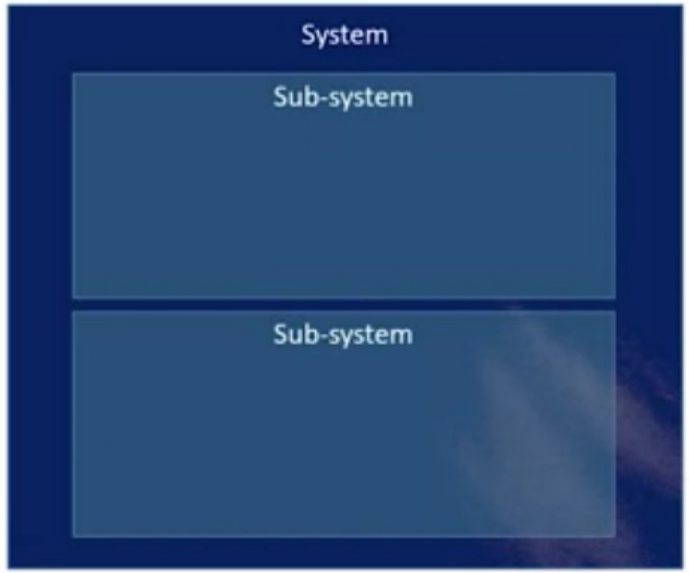
Module testing

Sub-system testing

System testing

Acceptance testing





Many software testing perspectives exist

Each has its own benefits

Use them all to get a better picture of how testing works

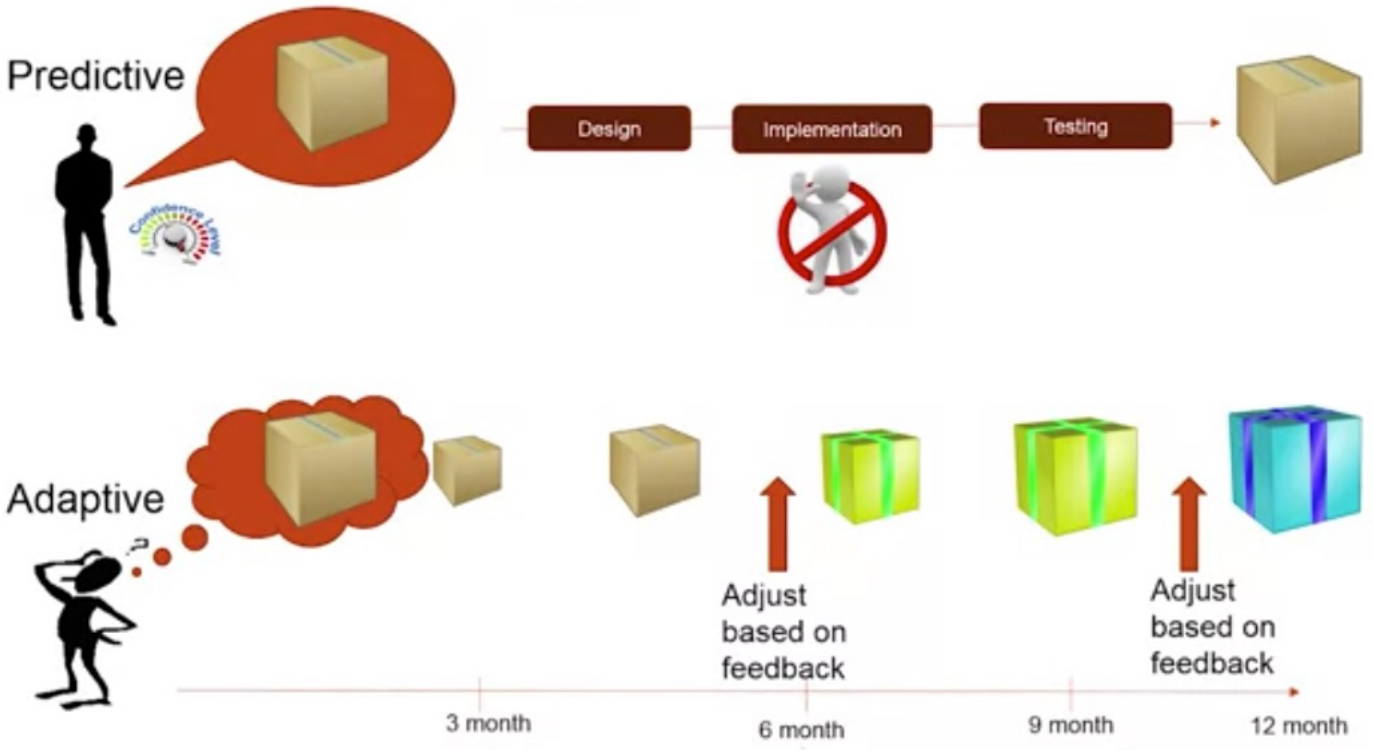
It is never one or the other; just use each in isolation

Even all of them combined will likely not be sufficient

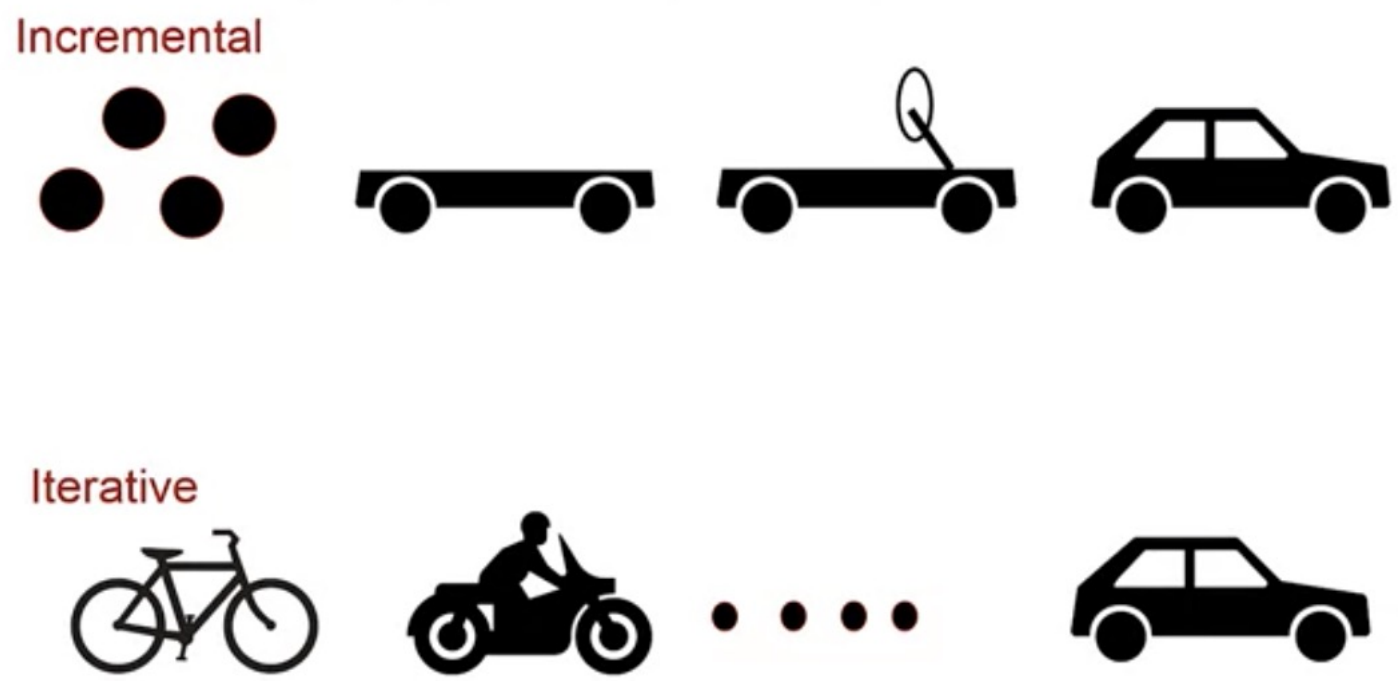
SOFTWARE DEVELOPMENT MODELS

- No Silver Bullet !!

PREDICTIVE VS ADAPTIVE



INCREMENTAL VS ITERATIVE



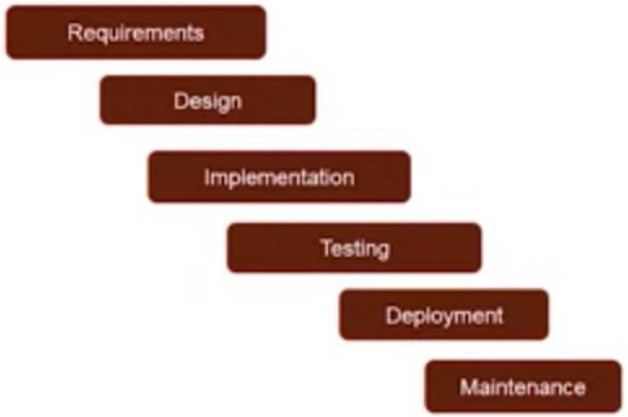
Both or None

<MODEL NAME>

- Information, Characteristics of the model

USE

- Situation in which you can/should use this model

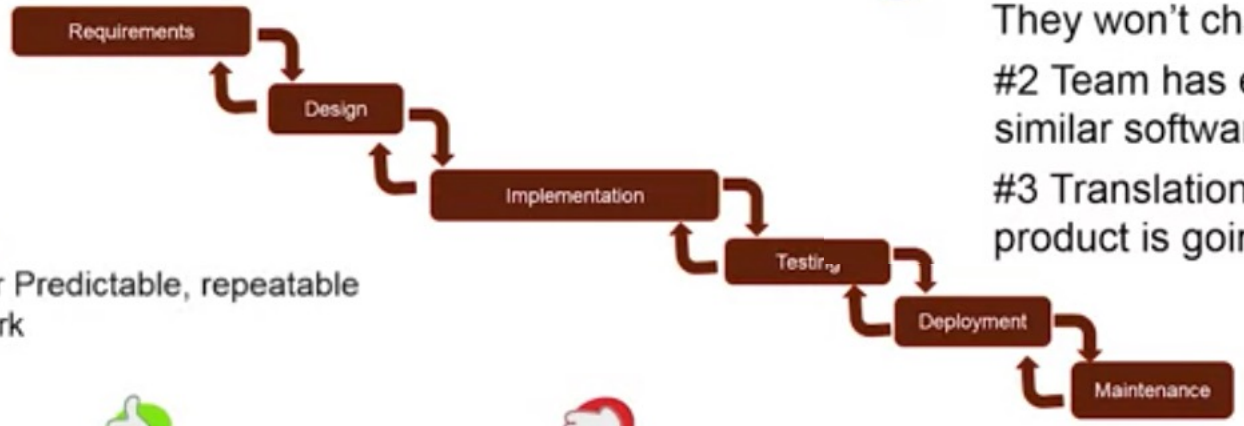


- Pros / Advantages of this approach

- Cons / Disadvantages of this model, approach

Where does this model stand in predictive vs adaptive scale.

WATERFALL MODEL



#1 We know requirements very well. They won't change.

#2 Team has experience building similar software.

#3 Translation from requirement to product is going to be perfect

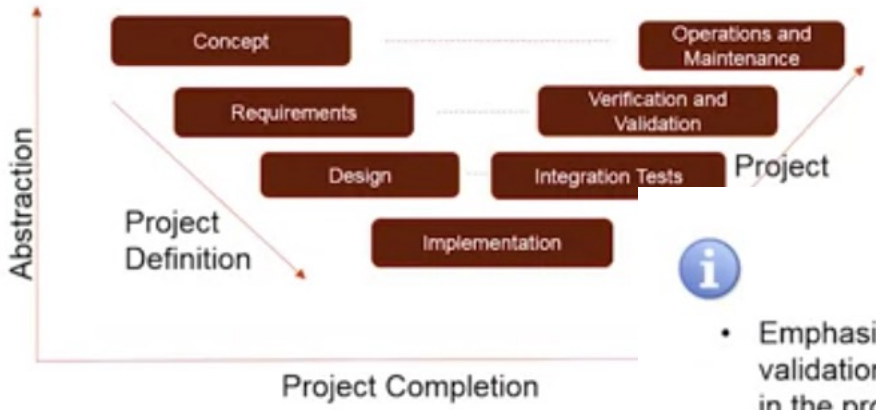
USE

- For Predictable, repeatable work

- Simple, Easy to understand
- Predictability
- Efficient

- Not flexible for change
- First release takes a long time

V-MODEL



- Emphasis on validation earlier in the process

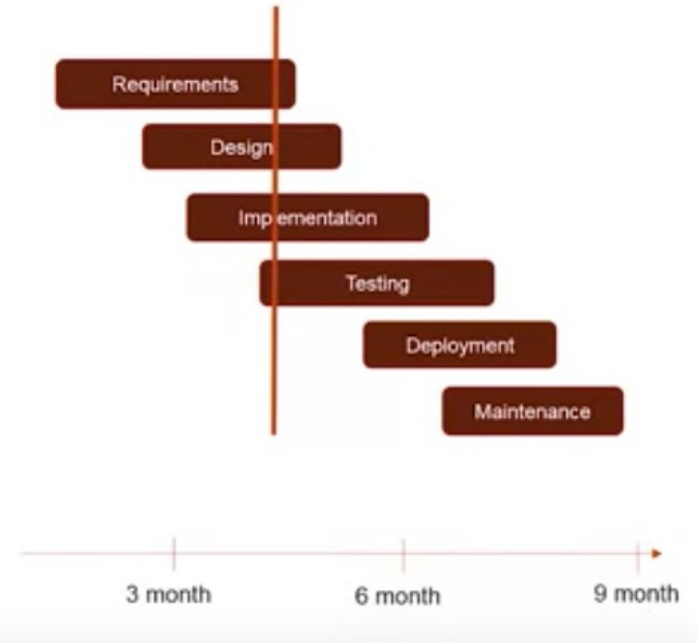
USE

- Ambiguity in requirements and early validation would be useful.

- Earlier detection of potential defects/issues
- More upfront work

Predictive — Adaptive

SASHIMI MODEL



- A phase can start before previous phase ends

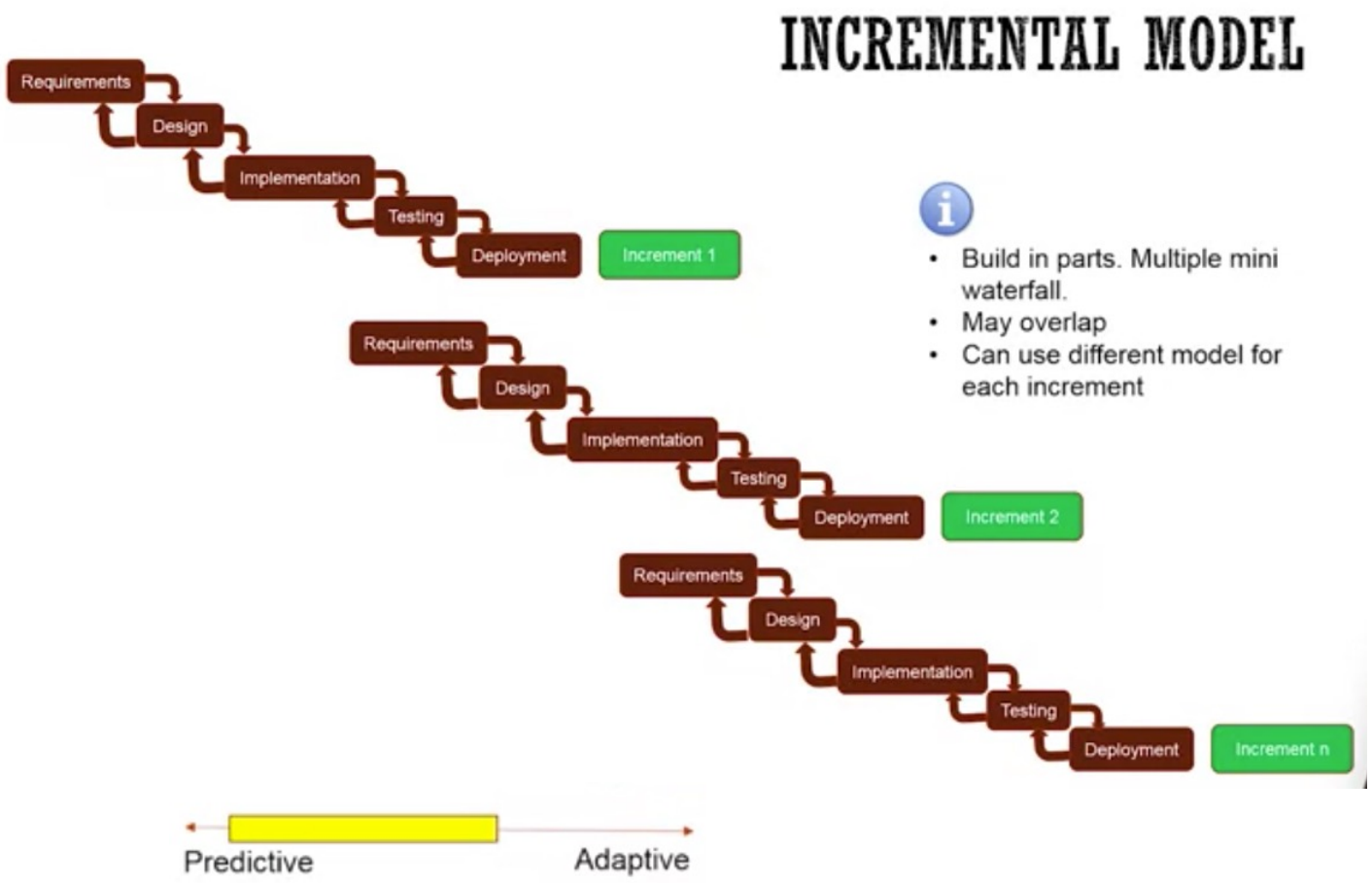
USE

- To shorten the time
- For resource utilization

- Shorten the development time
- People with different skill can start working without waiting
- Can do a learning spike early

- May result in rework

INCREMENTAL MODELS



- Build in parts. Multiple mini waterfall.
- May overlap
- Can use different model for each increment

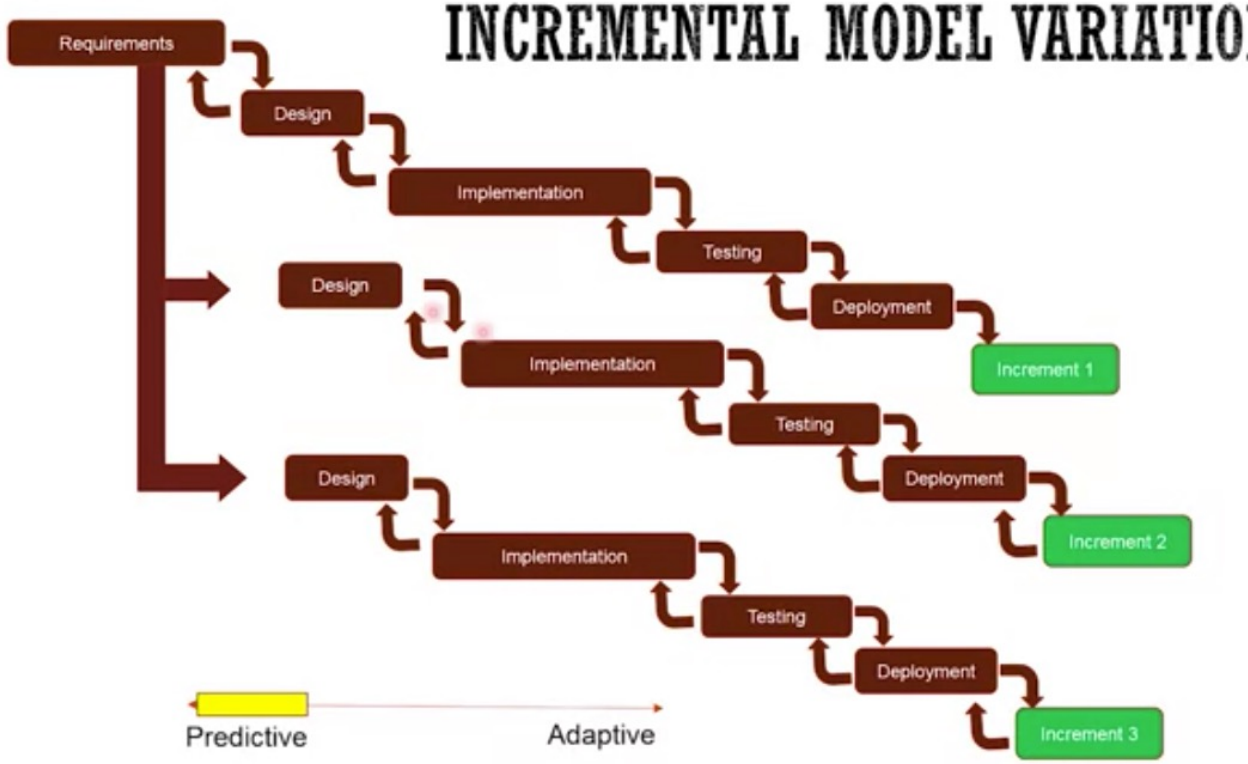
USE

- If organization may benefit from early delivery of part of product
- If building one increment will help define future increments

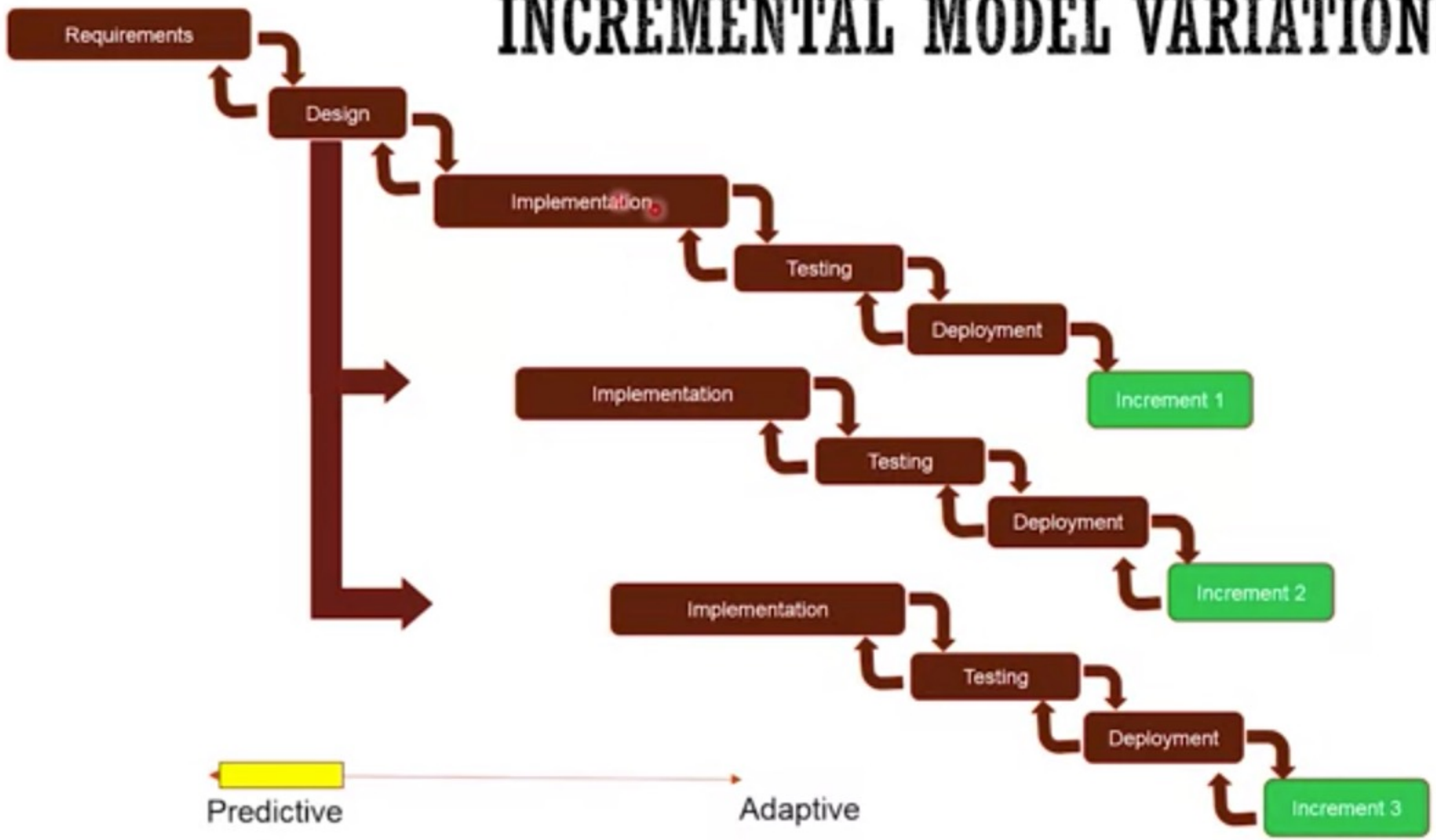
- Deliver value earlier
- Get feedback and make necessary changes between increments

- May result in rework.
- May cost more.

INCREMENTAL MODEL VARIATIONS

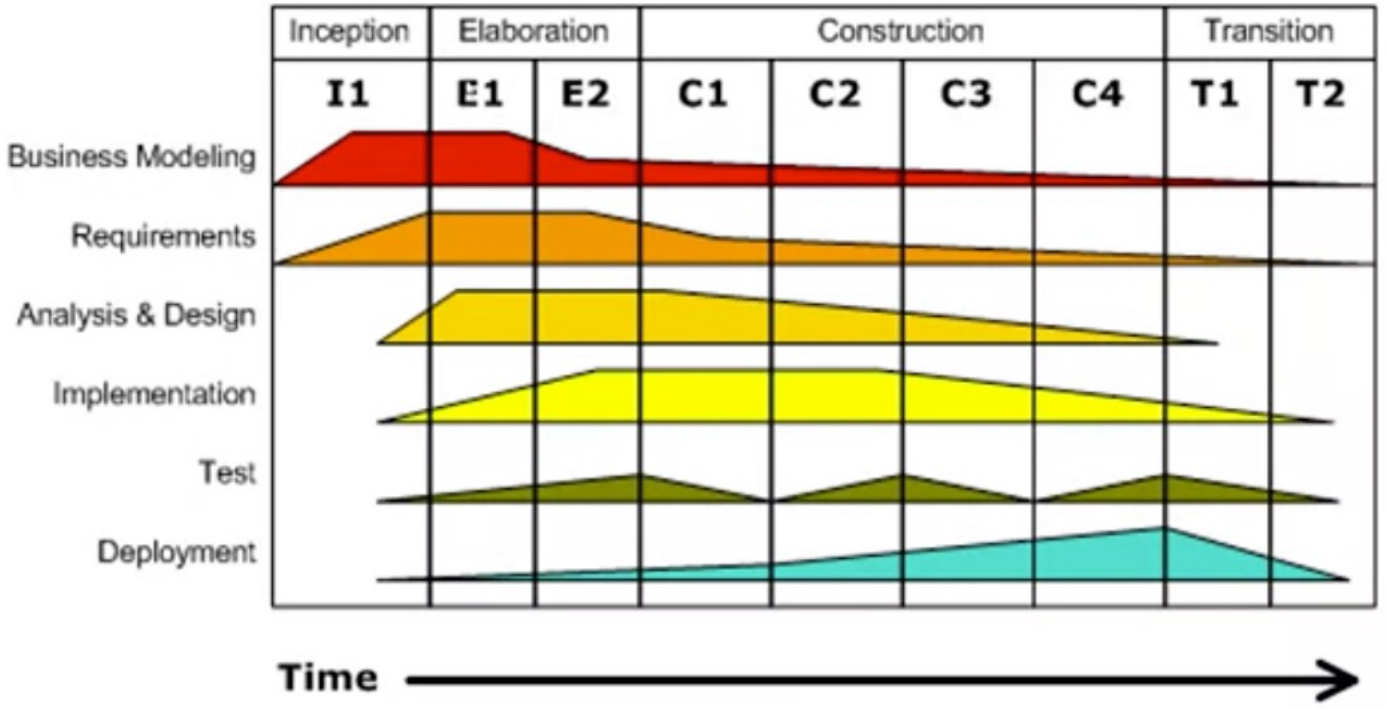


INCREMENTAL MODEL VARIATIONS



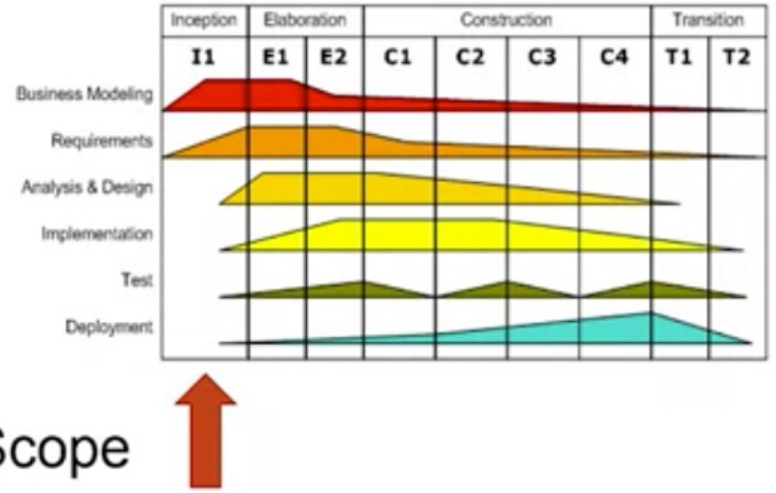
ITERATIVE MODELS

UNIFIED PROCESS MODEL



UNIFIED PROCESS AND ITS VARIANTS

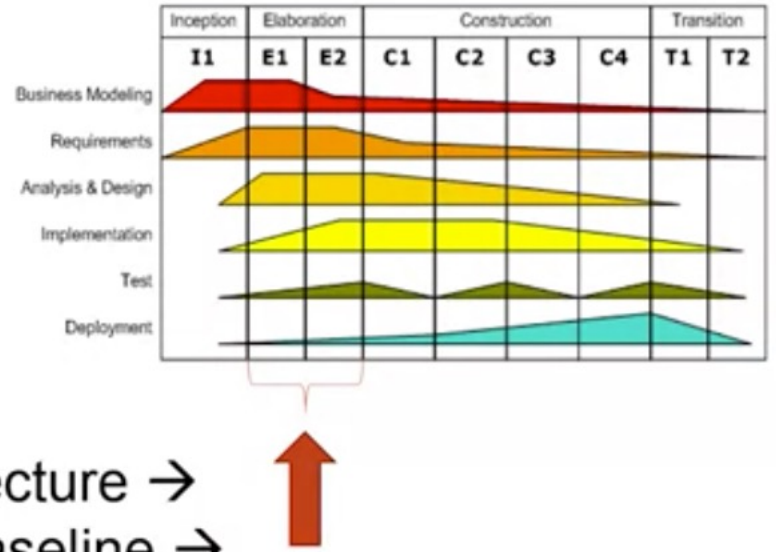
INCEPTION



- Short Phase
- What happens?
  - Establish Business case, Scope
  - Feasibility
  - Build vs Buy
  - Preliminary Schedule and Cost
- Milestone: Lifecycle Objective

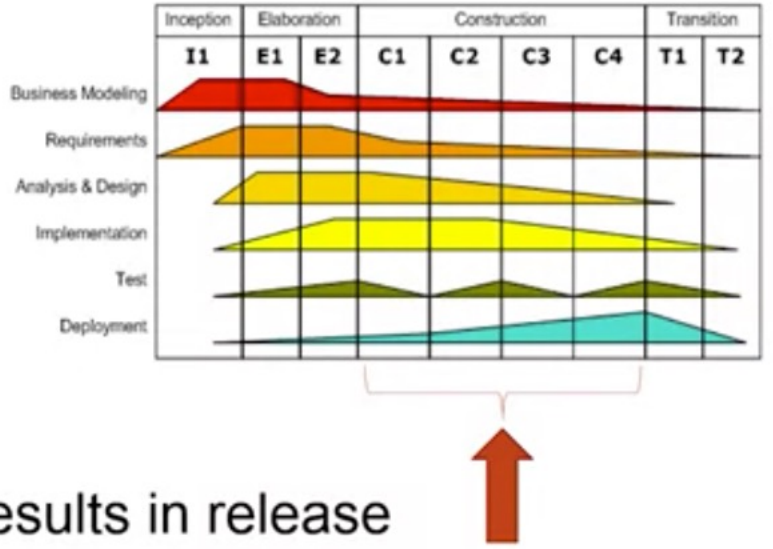
Time

ELABORATION



- What happens?
  - Capture requirements
  - Address known risks
  - Validate the system architecture → Executable Architecture Baseline →
  - Credible Construction Estimates
- Milestone: Lifecycle Architecture

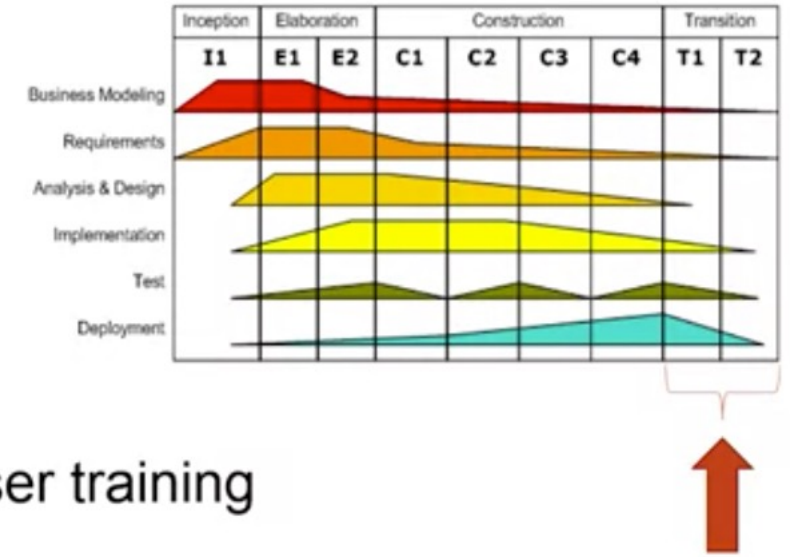
CONSTRUCTION



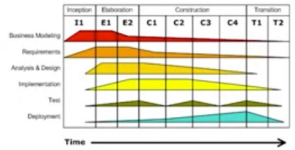
- Largest Phase
- What happens?
  - Software is built
  - Multiple iterations – each results in release
  - Iterative and incremental
- Milestone: Initial Operational Capability

TRANSITION

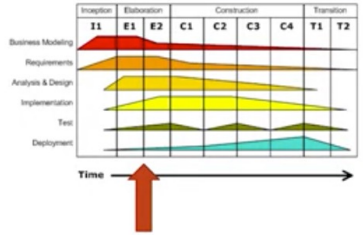
- What happens?
  - Deployment
  - Get feedback and refine
  - System conversion and user training



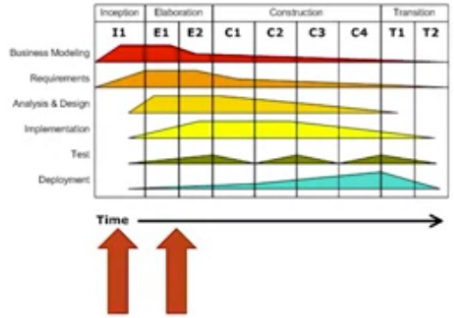
CHARACTERISTICS



ANY STEP MAY INVOLVE DIFFERENT KINDS OF WORK (REQUIREMENTS, DESIGN ETC) BUT RELATIVE EFFORT AND EMPHASIS MIGHT BE DIFFERENT



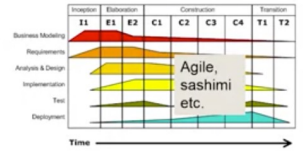
ARCHITECTURE CENTRIC



FOCUS ON RISK MITIGATION

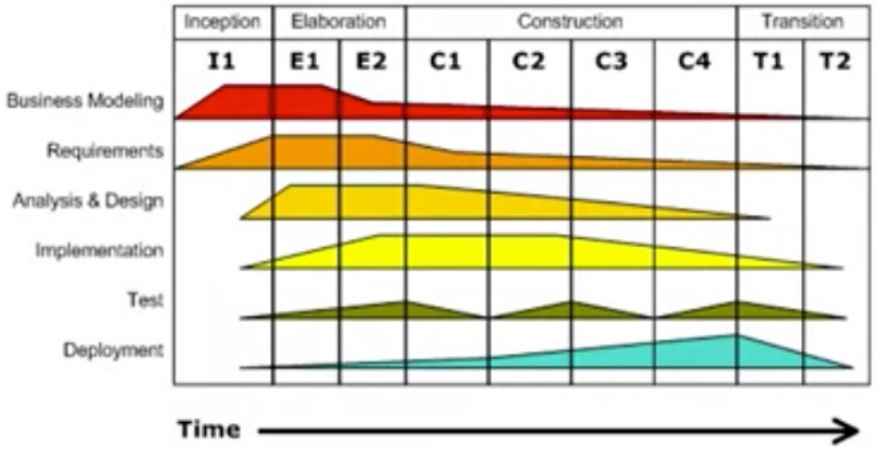
USE CASE #	< the name is the goal as a short active verb phrase>	
Goal in Context	<a longer statement of the goal in context if needed>	
Scope & Level	<what system is being considered black box under design> <one of: Summary, Primary Task, Sub-function>	
Preconditions	<what we expect is already the state of the world>	
Success End Condition	<the state of the world upon successful completion>	
Failed End Condition	<the state of the world if goal abandoned>	
Primary, Secondary Actors	<a role name or description for the primary actor>. <other systems relied upon to accomplish use case>	
Trigger	<the action upon the system that starts the use case>	
Description	Step	Action
	1	<put here the steps of the scenario from trigger to goal delivery, and any cleanup after>
	2	<...>
	3	
Extensions	Step	Branching Action
	1a	<condition causing branching> : <action or name of sub-use case>
Sub-Variations		Branching Action
	1	<list of variations>

USE CASE CENTRIC



IT IS A FRAMEWORK. IT INCORPORATES OTHER METHODS/MODELS.

UNIFIED PROCESS



USE

- Bigger and riskier projects
- All requirements not known early in the project
- Desire to deliver value earlier

- Adaptive
- Quality and Reuse
- Focus on risk mitigation increases chances of success
- Flexible to incorporate other s/w dev models

- Complicated
- Need more resources
- Too much overhead for smaller project

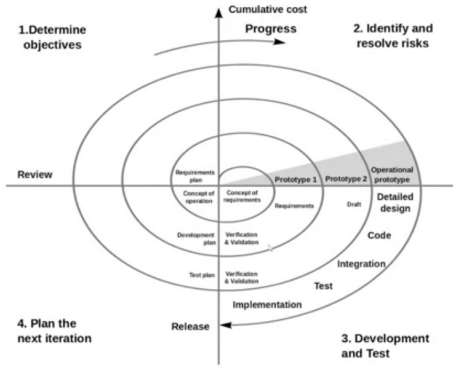
UNIFIED PROCESS VARIANTS

- Rational Unified Process – 9 Disciplines, 6 Best Practices + Tools
- Enterprise Unified Process
- Open UP --- Lighter version
- Agile UP --- Lighter version, Agile focused

SPIRAL MODEL

WHAT DOES IT LOOK LIKE?

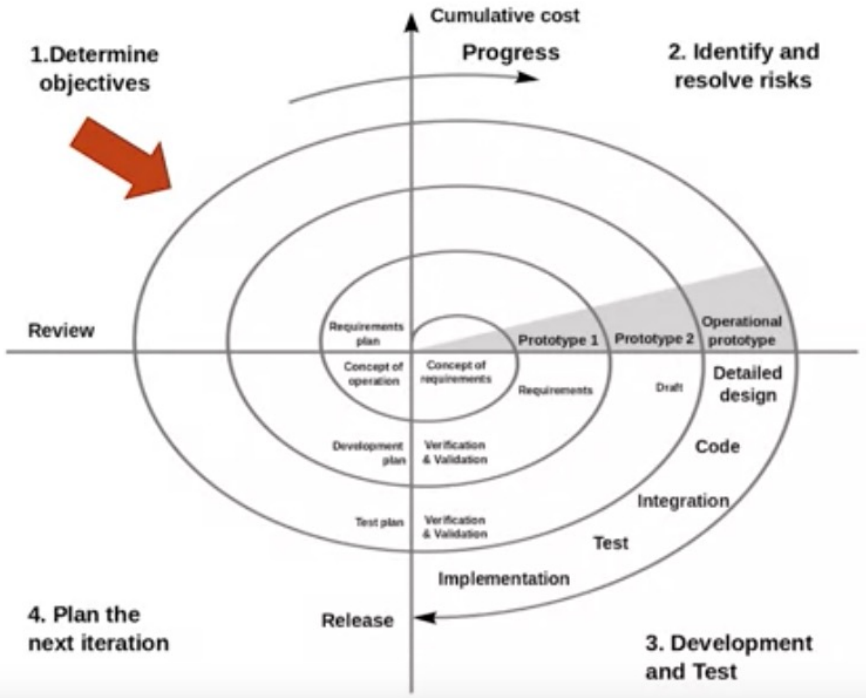
1. Cyclic
2. Four basic steps
3. Not every activity need to be performed



SPIRAL MODEL

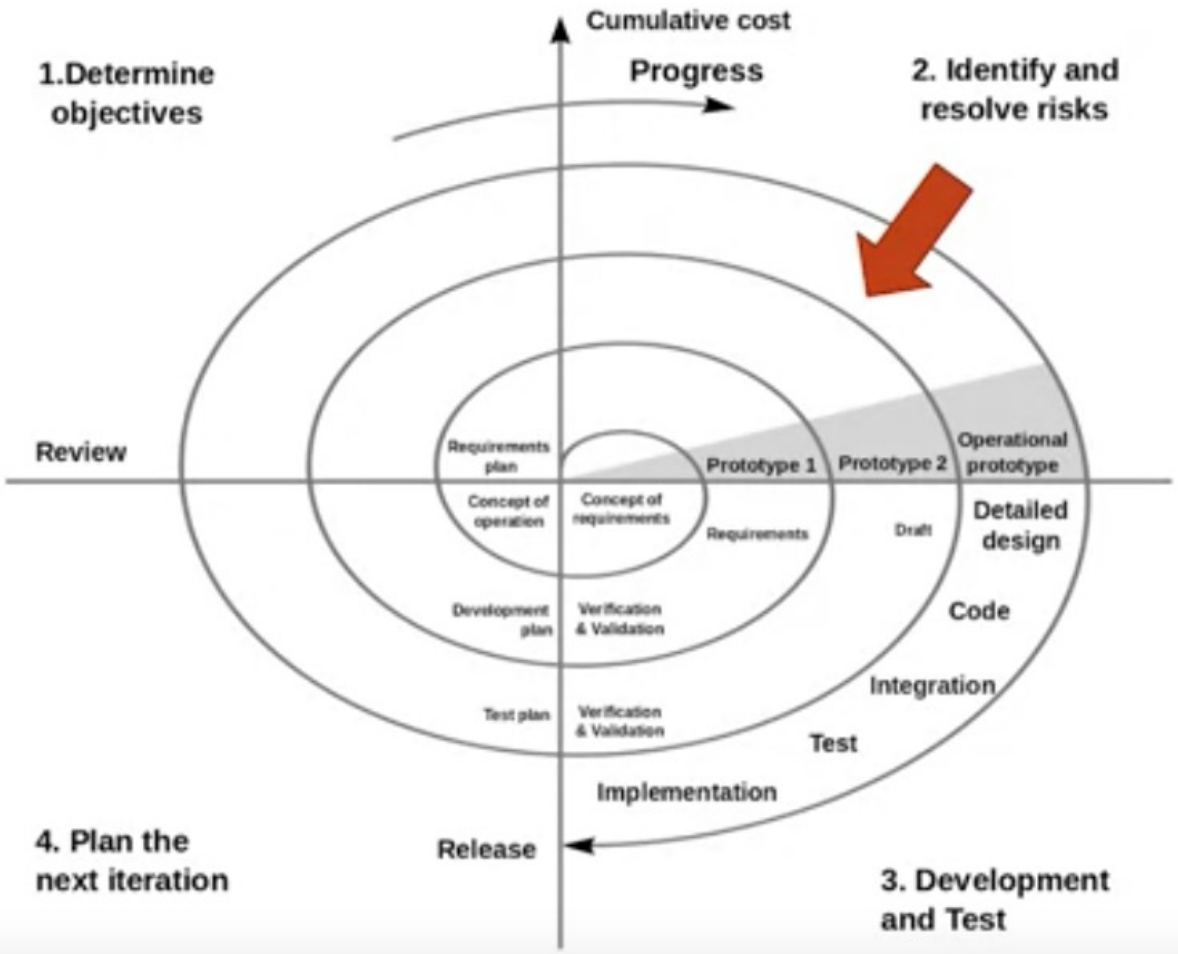
DETERMINE OBJECTIVES

1. Objectives
2. Constraints
3. Alternatives



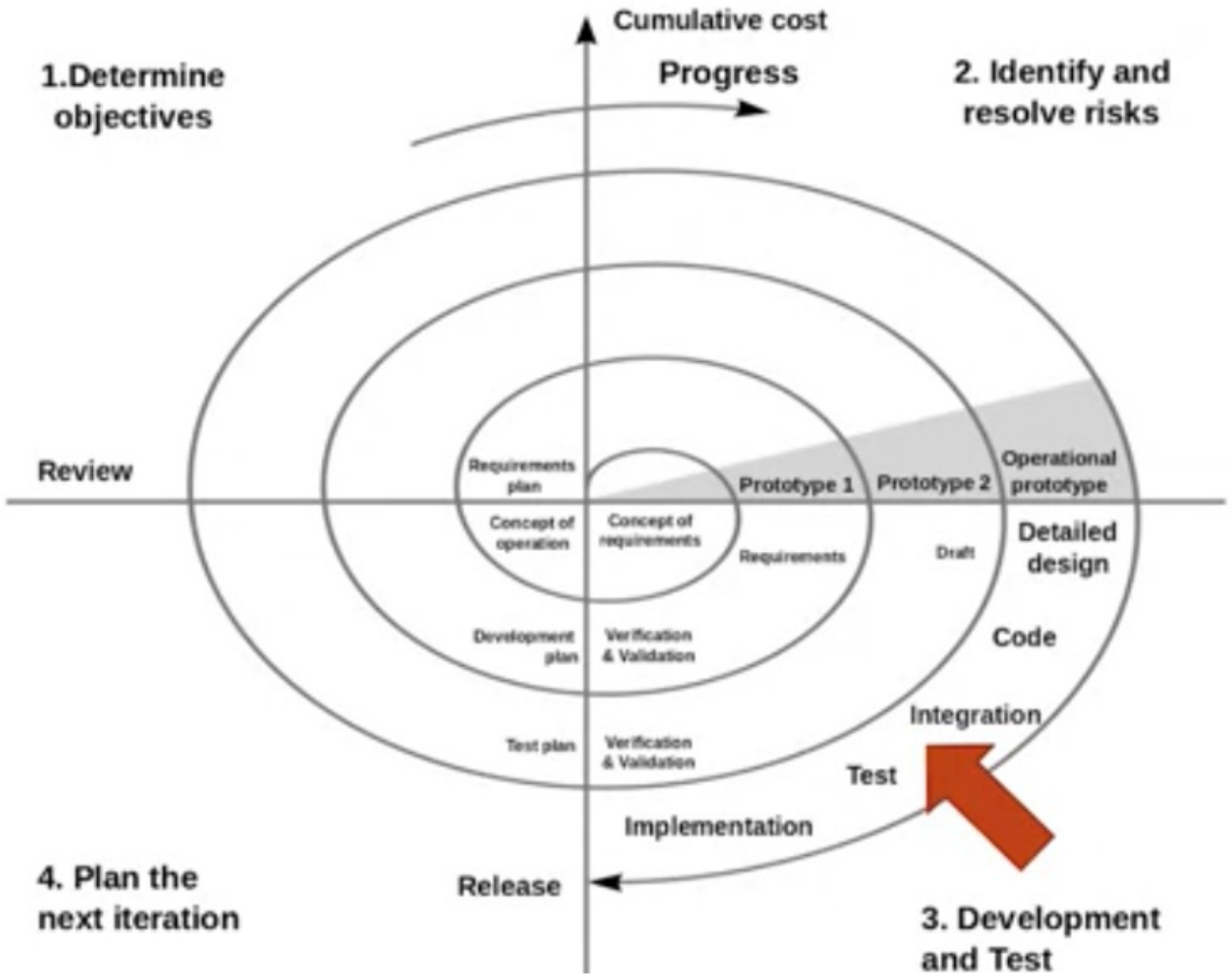
IDENTIFY AND RESOLVE RISKS

1. Identify Risks
2. Resolve
   what you can



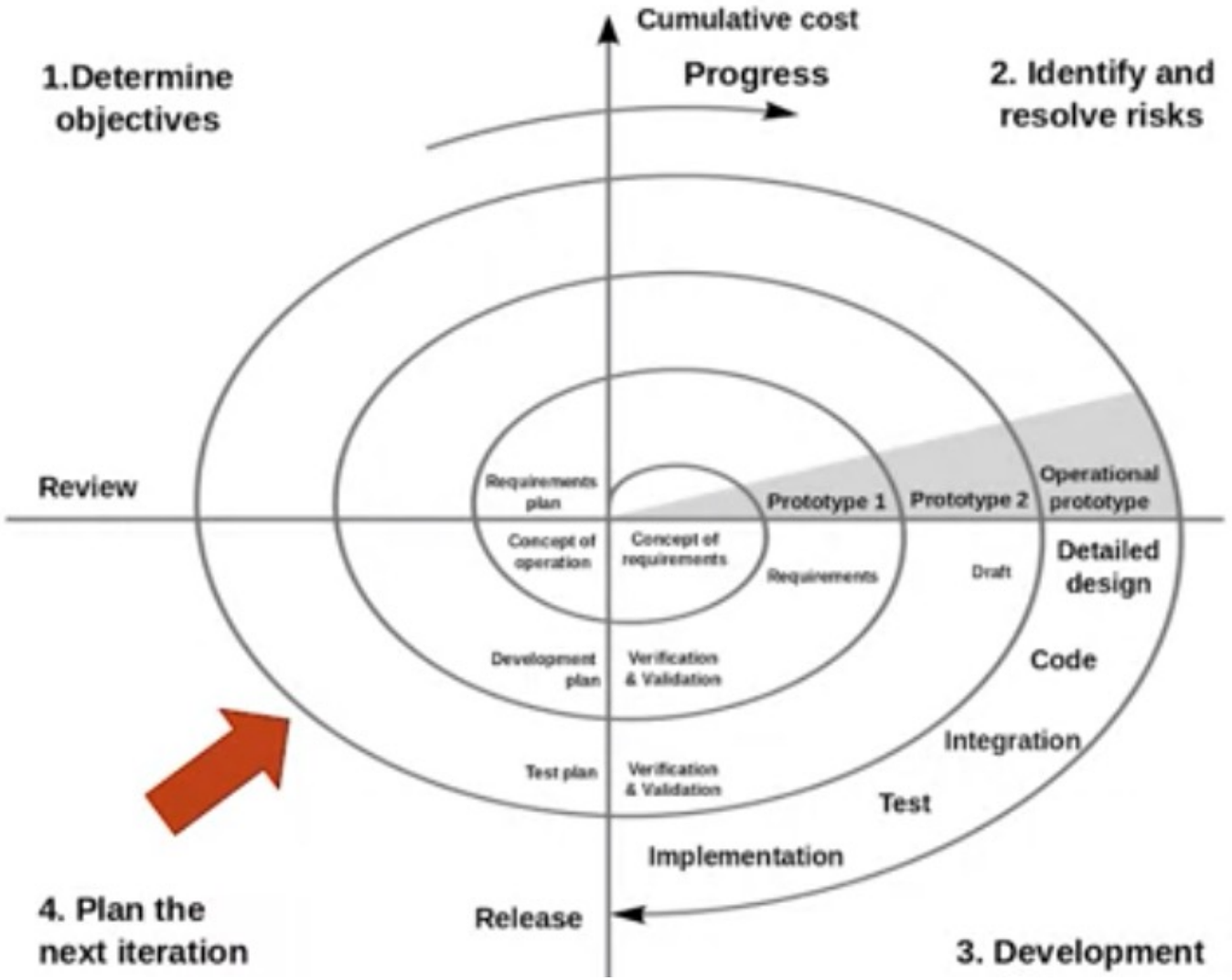
DEV AND TESTS

Work done to meet objectives



PLAN FOR NEXT ITERATION

Review work done and commitment for next iteration



Life Cycle Objective: sufficient definiti
of a technical and management approa

Life Cycle Architecture: sufficient
definition of the preferred approach +
significant risks eliminated or mitigated

Initial Operational Capability: sufficie
preparation of the software, site, users,
operators, and maintainers

HOW TO TRACK PROGRESS

KEY CHARACTERISTICS

- Risk driven model
- Effort and Detail Driven by the Risk

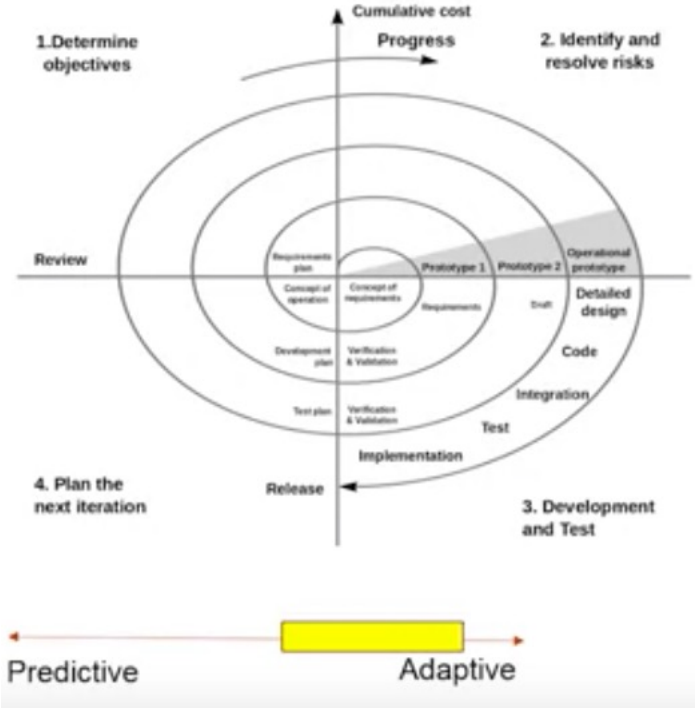
SPIRAL MODEL

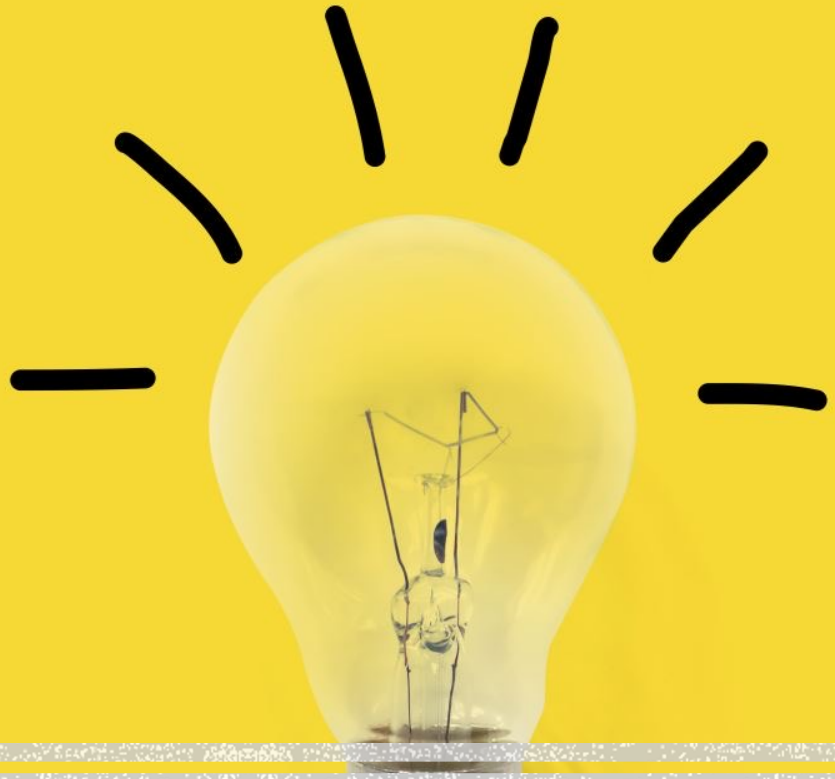
USE

- Very large High risk projects

- Adaptive
- Risk focus increases chances of success
- Flexible for using any model
- Minimizes waste.
- Options for go/no-go

- Complicated
- Cost more to manage
- Need stakeholder engagement



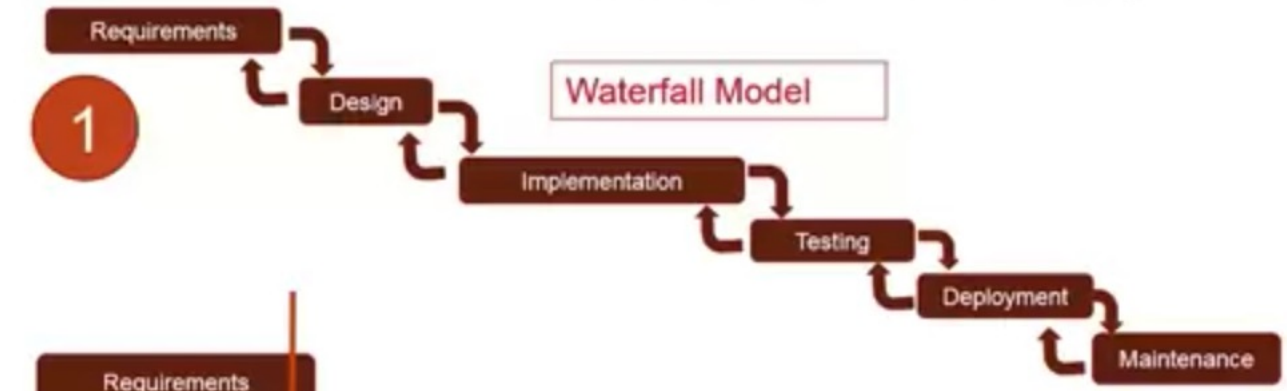


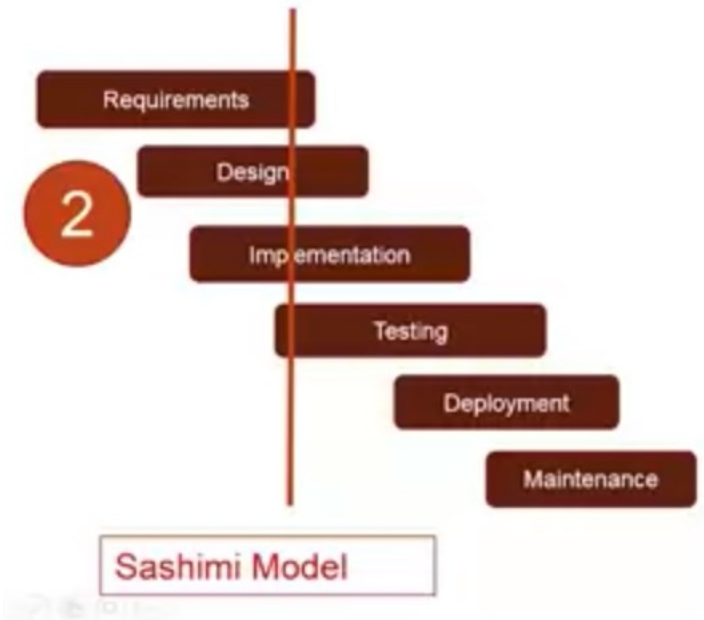
TIME TO TEST YOUR LEARNING !!

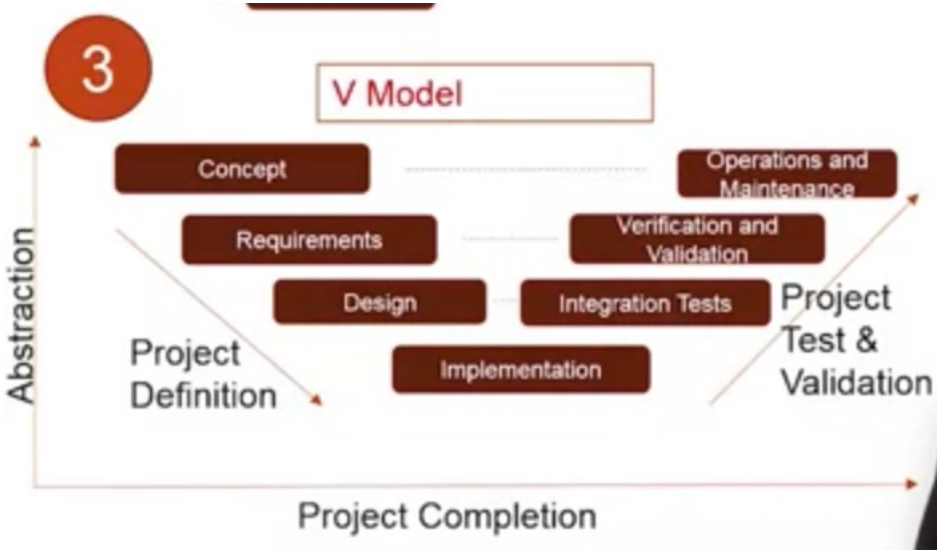
EXAMPLE 1

- Company X need to install a well known HR Management System at a big retailer's HQ.
- Company X has done many such installs on other big retailers before.

KNOWN PROBLEM, KNOWN SOLUTION







EXAMPLE 2

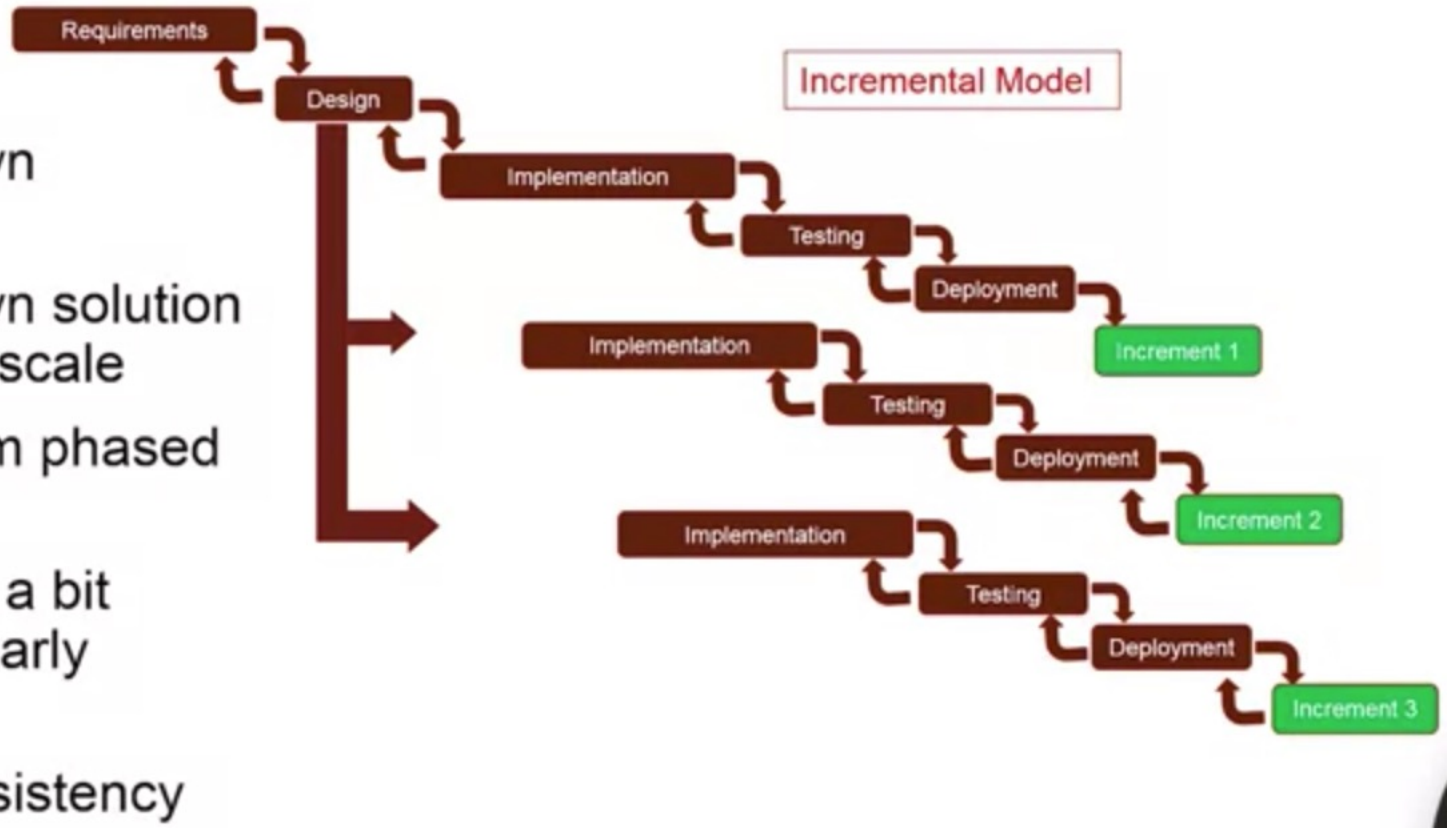
- Very large hospital with locations all over the world wanted to automate their processes
- Hired a company that is expert in this area and has done similar automation but not at this scale.
- Hospital management want consistency across the globe but not sure about the nuances in different part of the world.
- Few of the places can benefit greatly from the immediate automation

EXAMPLE 2 – ANALYSIS & RECOMMENDATIONS

- Fairly known problem
- Fairly known solution except the scale
- Benefit from phased delivery
- May tweak a bit based on early iterations
- Wants consistency

EXAMPLE 2 – ANALYSIS & RECOMMENDATIONS

- Fairly known problem
- Fairly known solution except the scale
- Benefit from phased delivery
- May tweak a bit based on early iterations
- Wants consistency

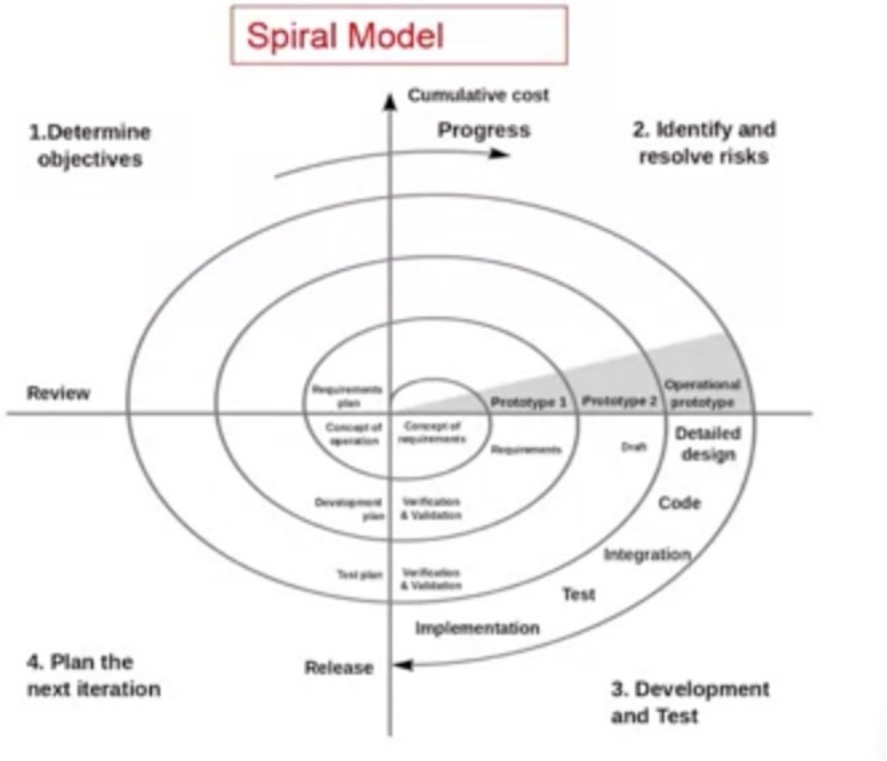


EXAMPLE 3

- Defense organization of a country did a recent study and the research recommends new capability the country should build to keep the country protected from potential conflicts in the region.
- The system required to be build has never been attempted and no literature exists for such system.
- It is fairly big and complex system and potentially can take a decade to build.
- Scientists have vague ideas but no concrete plan exists.
- There are lot of Organization stakeholders and constraints that will impact this initiative.

EXAMPLE 3 – ANALYSIS & RECOMMENDATIONS

- Fairly unknown needs and outcomes
- Very very risky
- Very large and complex project



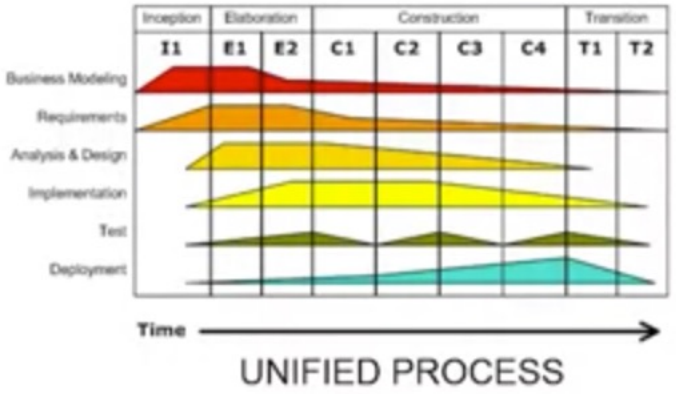
EXAMPLE 4

- A fairly big organization has a product for processing medical prior authorizations submitted by insurance members. Product has been in use for years. This Organization mostly follow traditional project mgmt.
- The current system has several limitations and has not been keeping up to date with current industry trends. It cannot fulfill functionality expected by clients.
- Organization wants to build a whole new system that will satisfy client needs and setup the organization for future.
- The potential team who will be working on this has fairly good domain knowledge and knows the base technology pretty well. There is still lot of new technology they will be working with.
- The current system is fairly complex and it won't be easy to migrate existing workflows to new system. Expected duration of this migration is around 2 years.
- Lot of stakeholders (including leaders, managers, potential users) will be impacted by this change. This means significant training, change management, stakeholder mgmt is needed to be successful.

EXAMPLE 4 - ANALYSIS

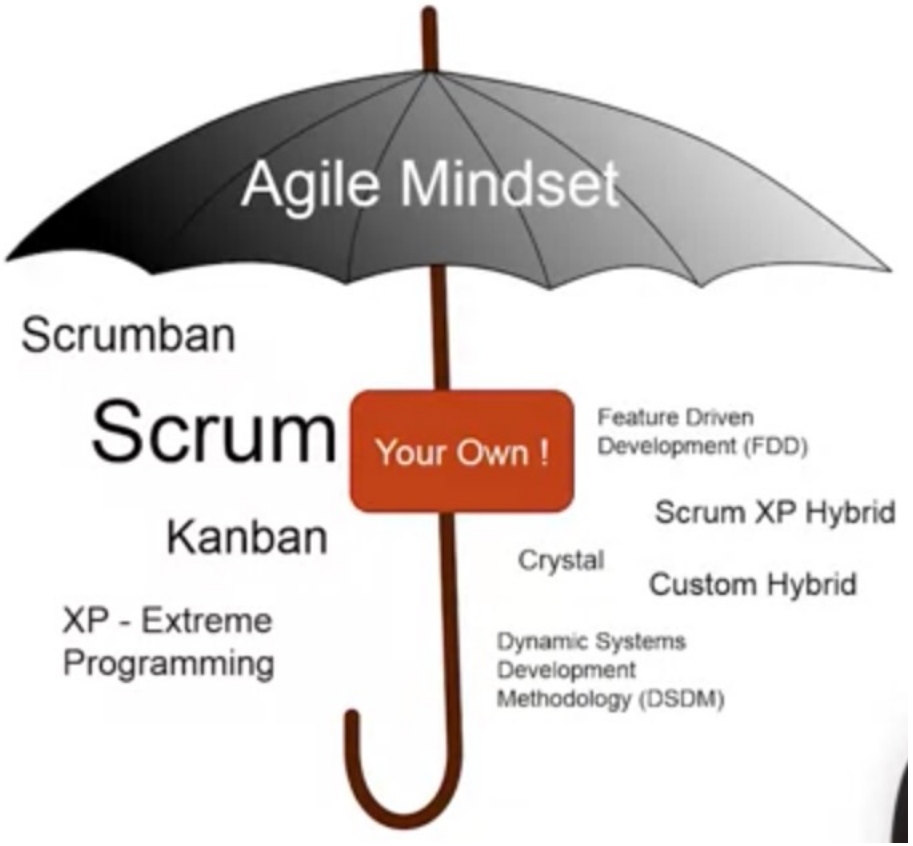
- Some unknowns and risks
- Since team has good domain knowledge and base technical expertise, it is not totally new space for the team.
- Medium size project
- Complex deployment and change management
- Architecture is key to setup the organization for future. Need some work to get this right.
- Looking at complexity and number of stakeholder, it will be beneficial to roll this out incrementally to get some feedback and tweak the system.
- Most of the projects in the Organization are managed using Traditional approaches and leaders prefer certainty, milestones and solid plans.

EXAMPLE 4 - RECOMMENDATIONS



UNIFIED PROCESS

OR



WHY AGILE?

NECESSITY IS THE MOTHER OF INVENTION!

IT IS NOT A SPECIFIC MODEL/PROCESS

AGILE IS A MINDSET!

MANIFESTO – 4 VALUES, 12 PRINCIPLES

http://en.wikipedia.org/wiki/Agile_software_development

12 PRINCIPLES

PRINCIPLES BEHIND AGILE MANIFESTO

1. Our highest priority is to **satisfy the customer through early and continuous delivery** of valuable software.
2. **Welcome changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
3. **Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. **Business people and developers must work together** daily throughout the project.
5. Build projects around **motivated individuals**. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation.**

PRINCIPLES BEHIND AGILE MANIFESTO

7. **Working software is the primary measure of progress**.
8. Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to **technical excellence** and good design enhances agility.- cost of exploration
10. **Simplicity**–the art of maximizing the amount of work not done–is essential.
11. The best architectures, requirements, and designs emerge from **self-organizing teams**.
12. At regular intervals, the **team reflects** on how to become more effective, then tunes and adjusts its behavior accordingly.

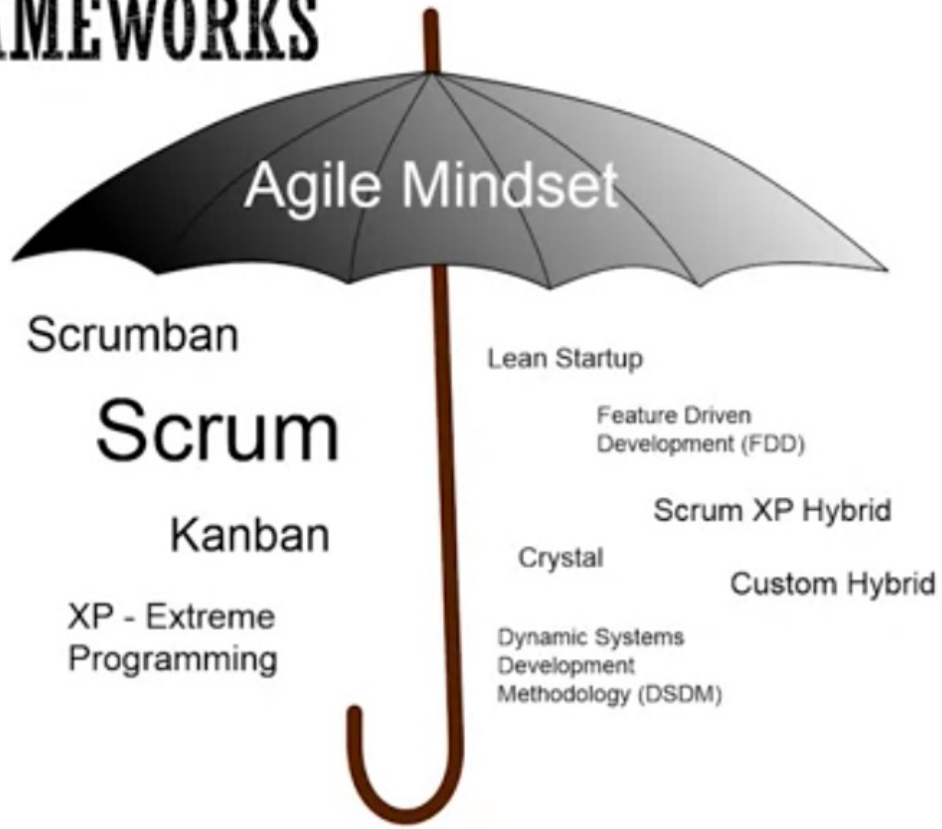
WHAT NEW PROBLEMS DOES IT BRING?

- Adaptive
  - Deliver working software frequently
  - Welcome change
  - Technical excellence and good design
  - Continuous improvement
- People and Interaction
  - Business and Developer work together
  - Face-to-Face conversations
  - Self organizing teams
  - Promote sustainable development
  - Motivated individuals

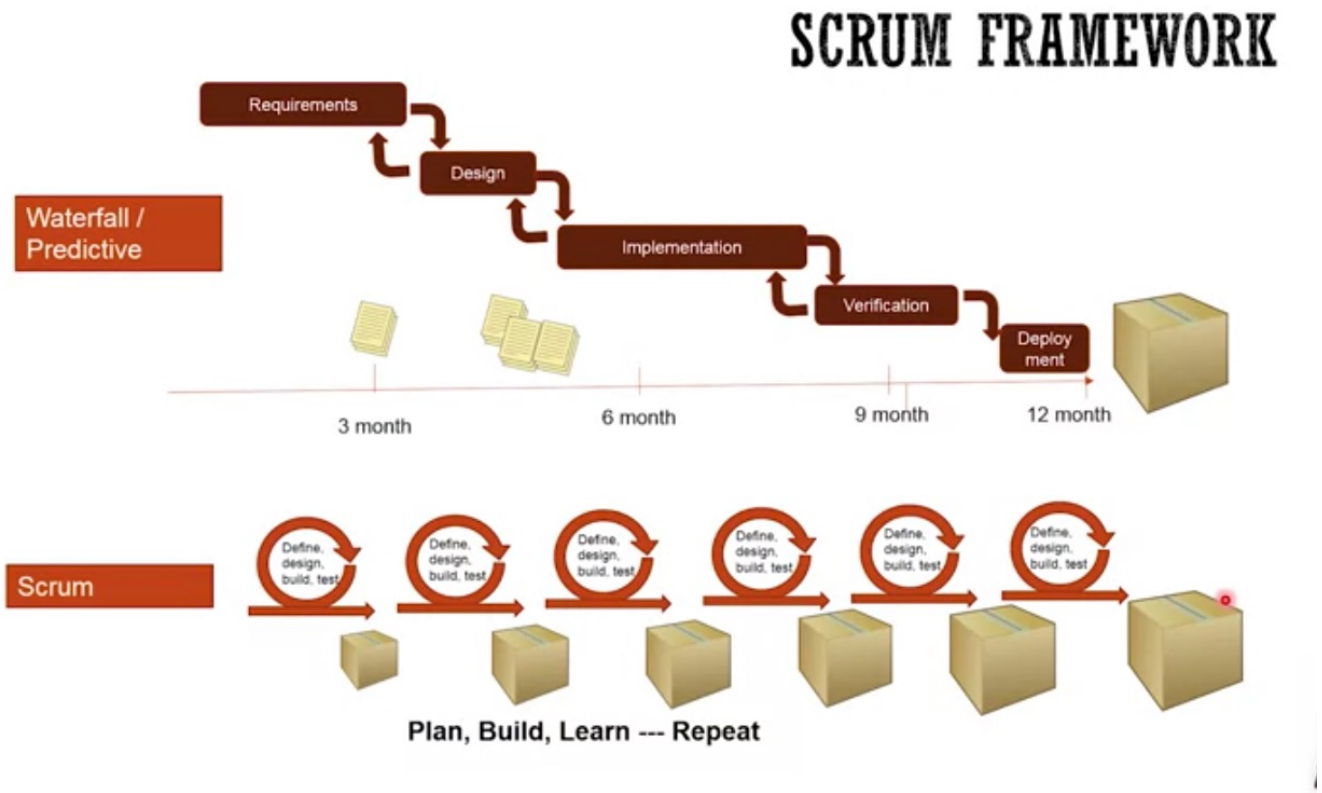
- Architecture/Design/Database modeling is challenging
- Lack of control / Unpredictable Journey - Very uncomfortable for Leaders/Organizations

- Requires participation from customers through out the development process

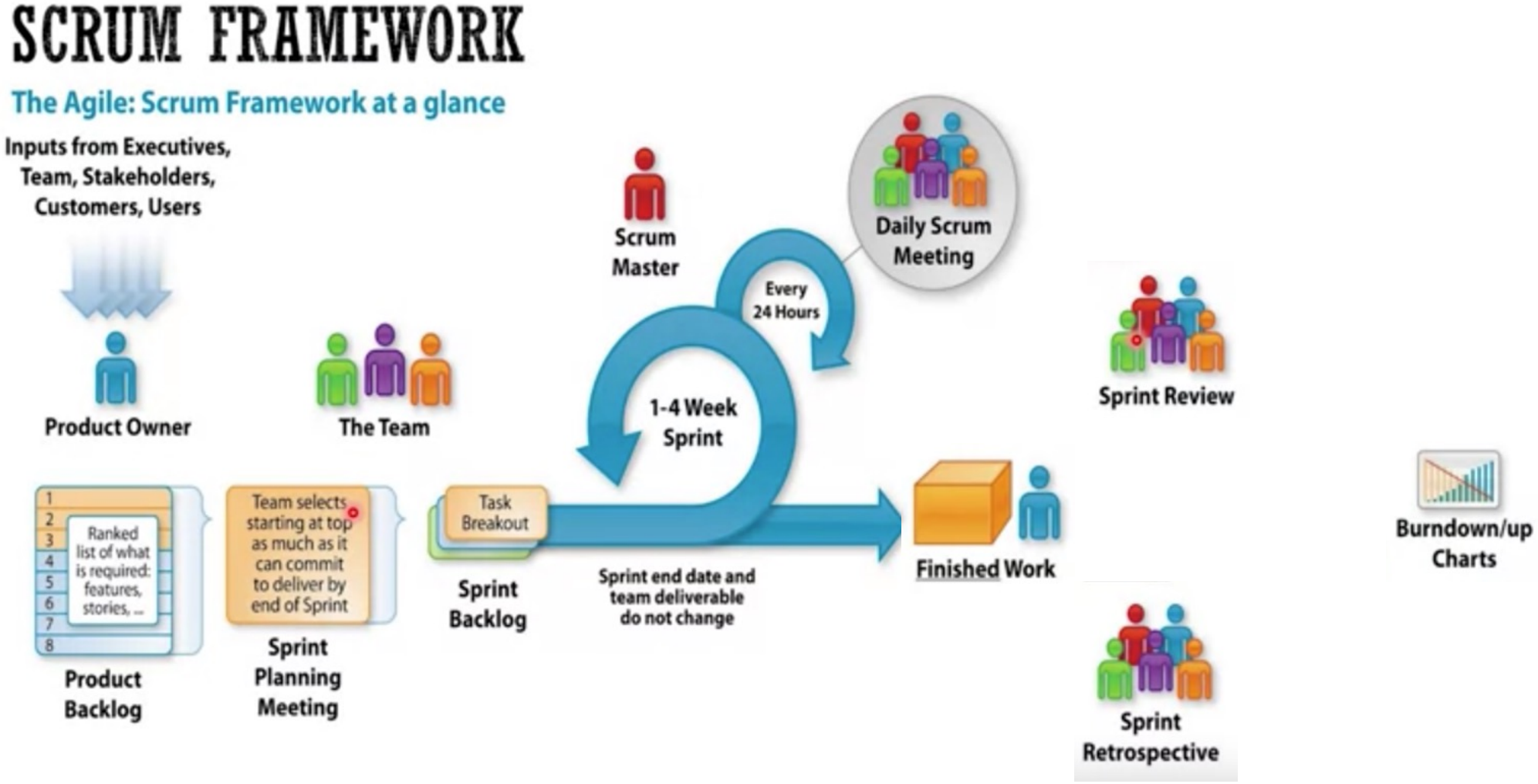
AGILE FRAMEWORKS



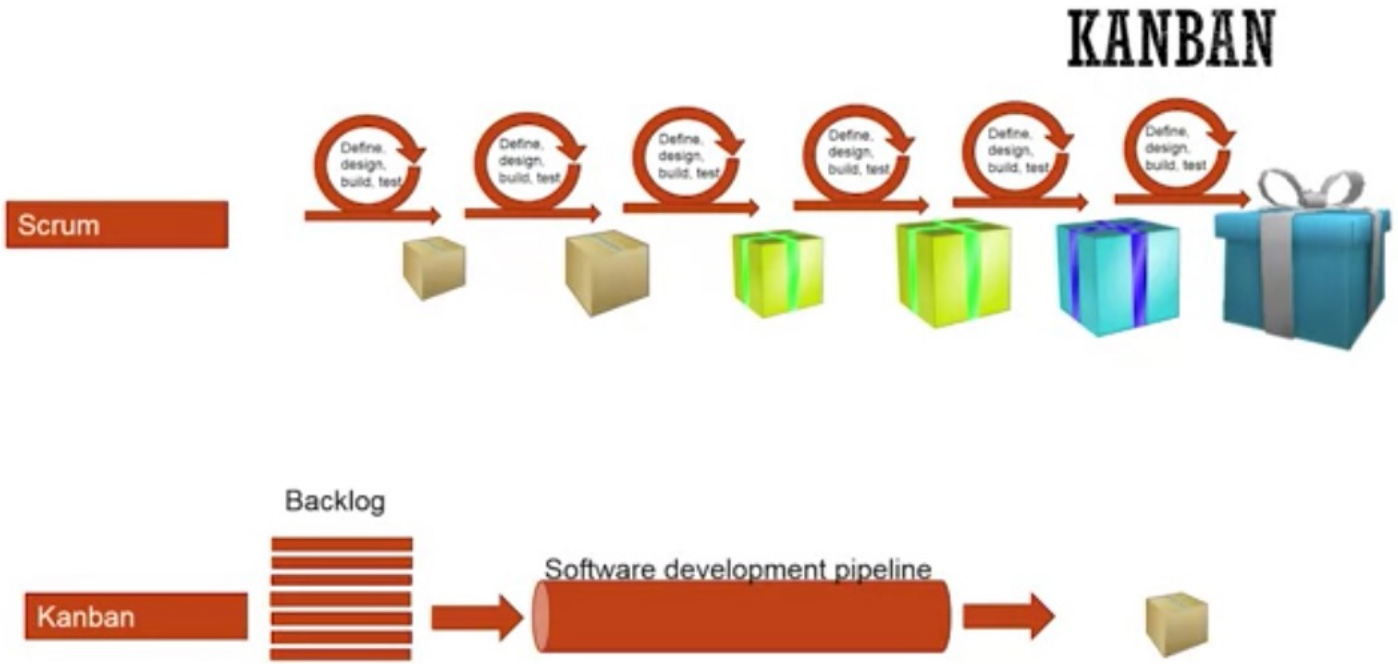
SCRUM


SCRUM FRAMEWORK
Waterfall / Predictive
Requirements
Design
Implementation
Verification
Deploy ment
3 month
6 month
9 month
12 month
Scrum
Define, design, build, test
Plan, Build, Learn --- Repeat

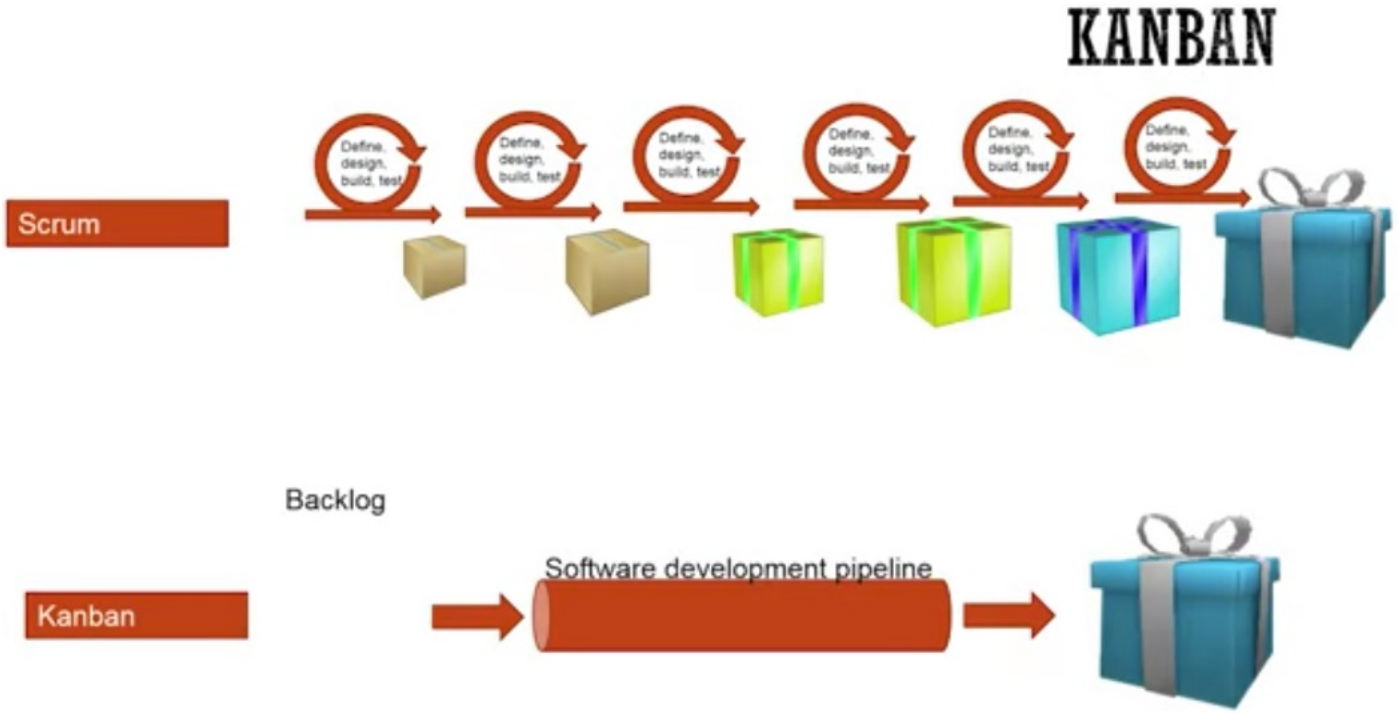
SCRUM FRAMEWORK



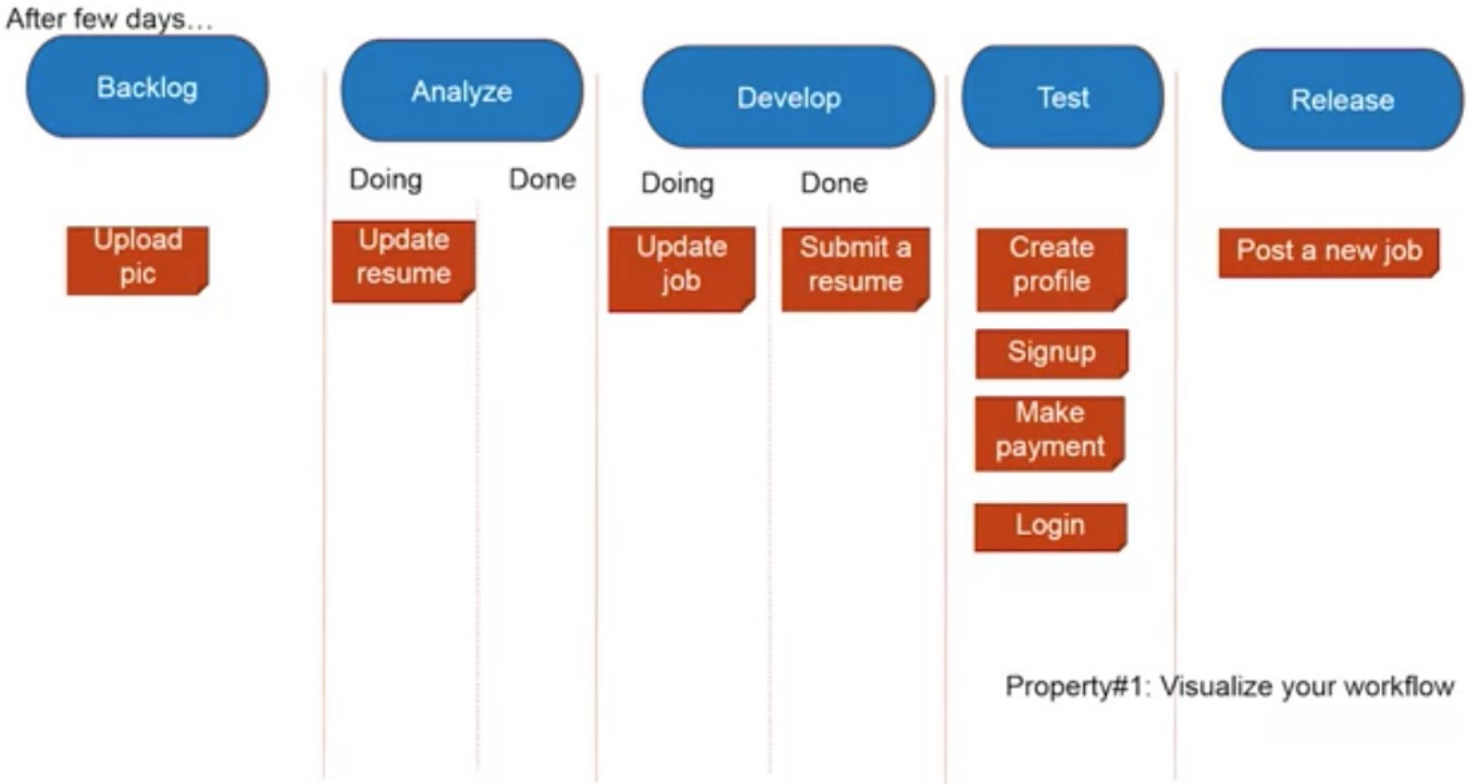
KANBAN


KANBAN
Scrum
Define, design, build, test
Backlog
Kanban
Software development pipeline

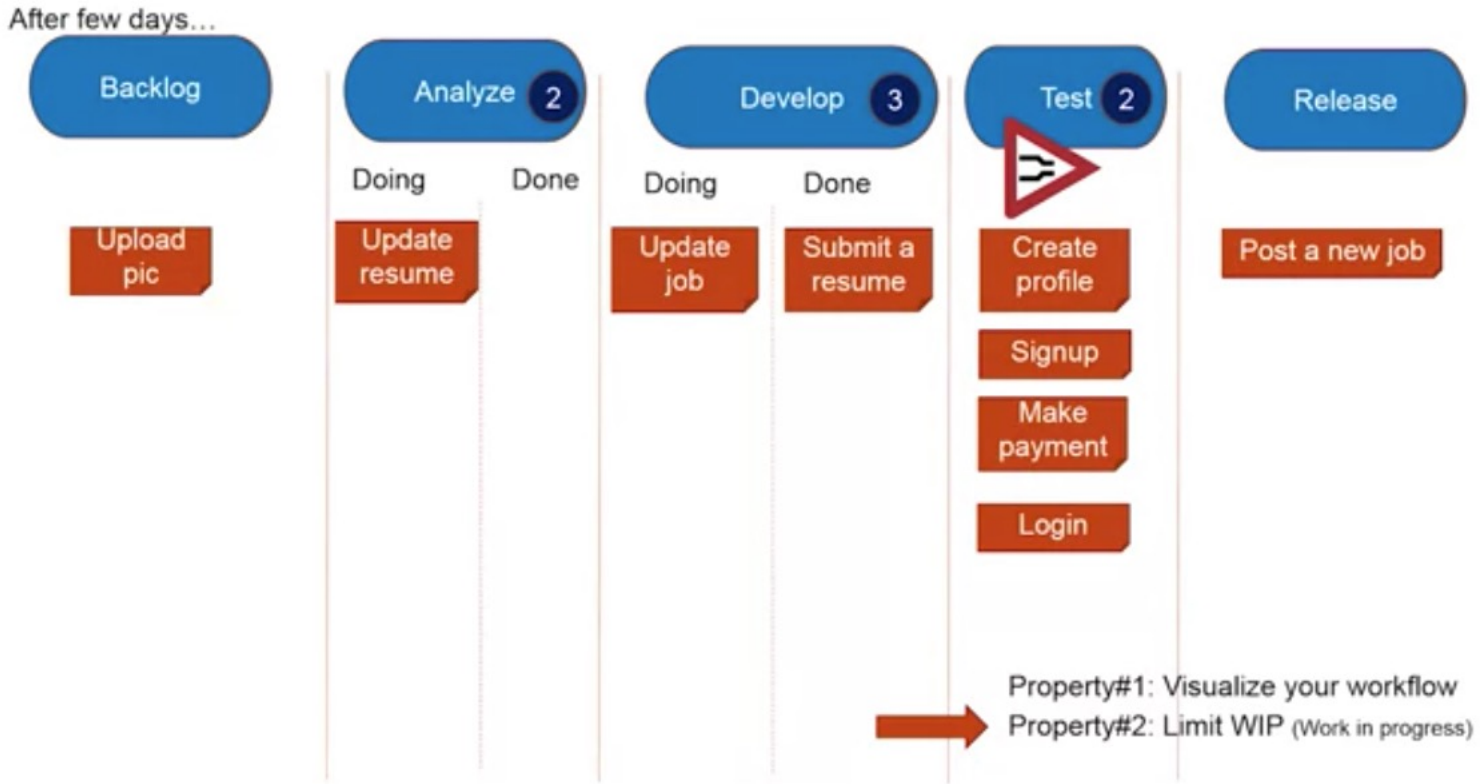
KANBAN


KANBAN
Scrum
Define, design, build, test
Backlog
Kanban
Software development pipeline

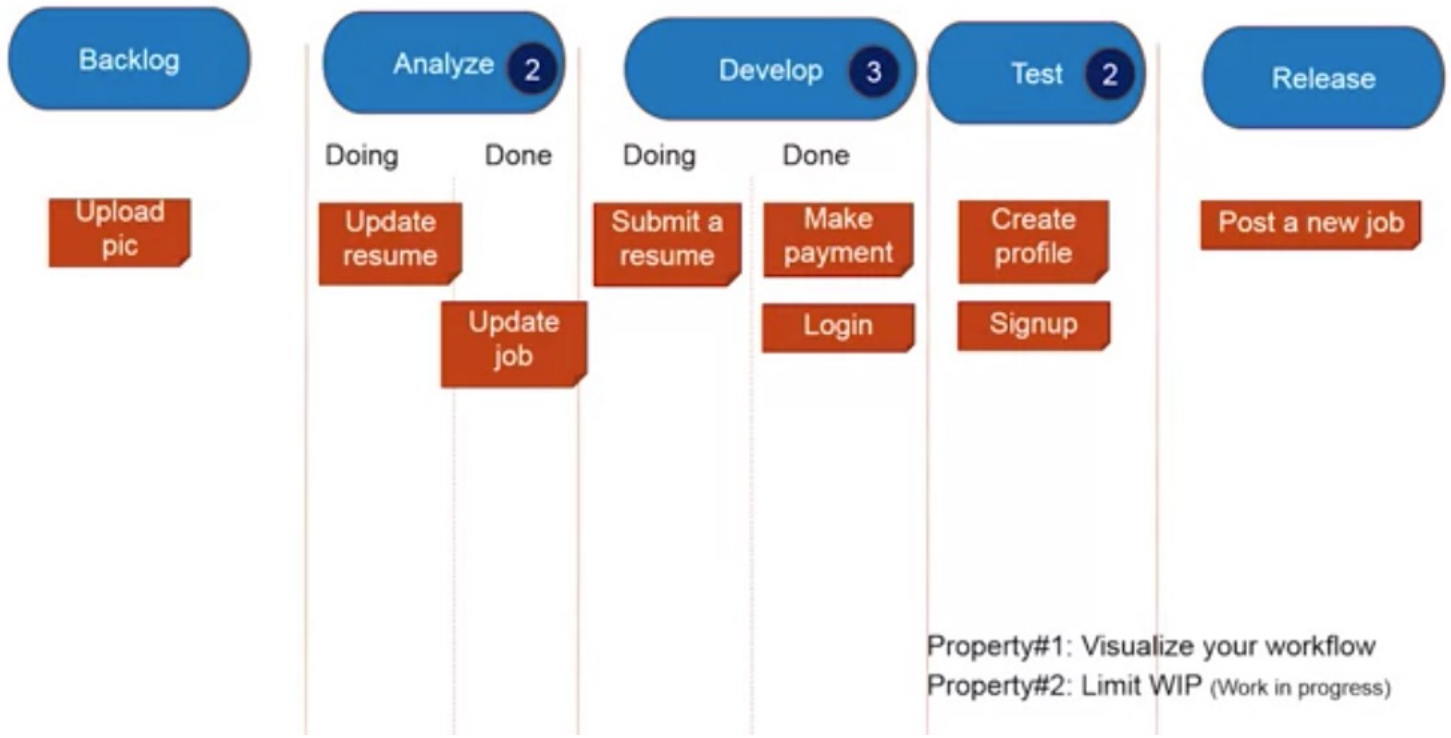
KANBAN


After few days…
Backlog
Analyze
Develop
Test
Release
Doing
Done
Doing
Done
Upload pic
Update resume
Update job
Submit a resume
Create profile
Signup
Make payment
Login
Post a new job
Property#1: Visualize your workflow

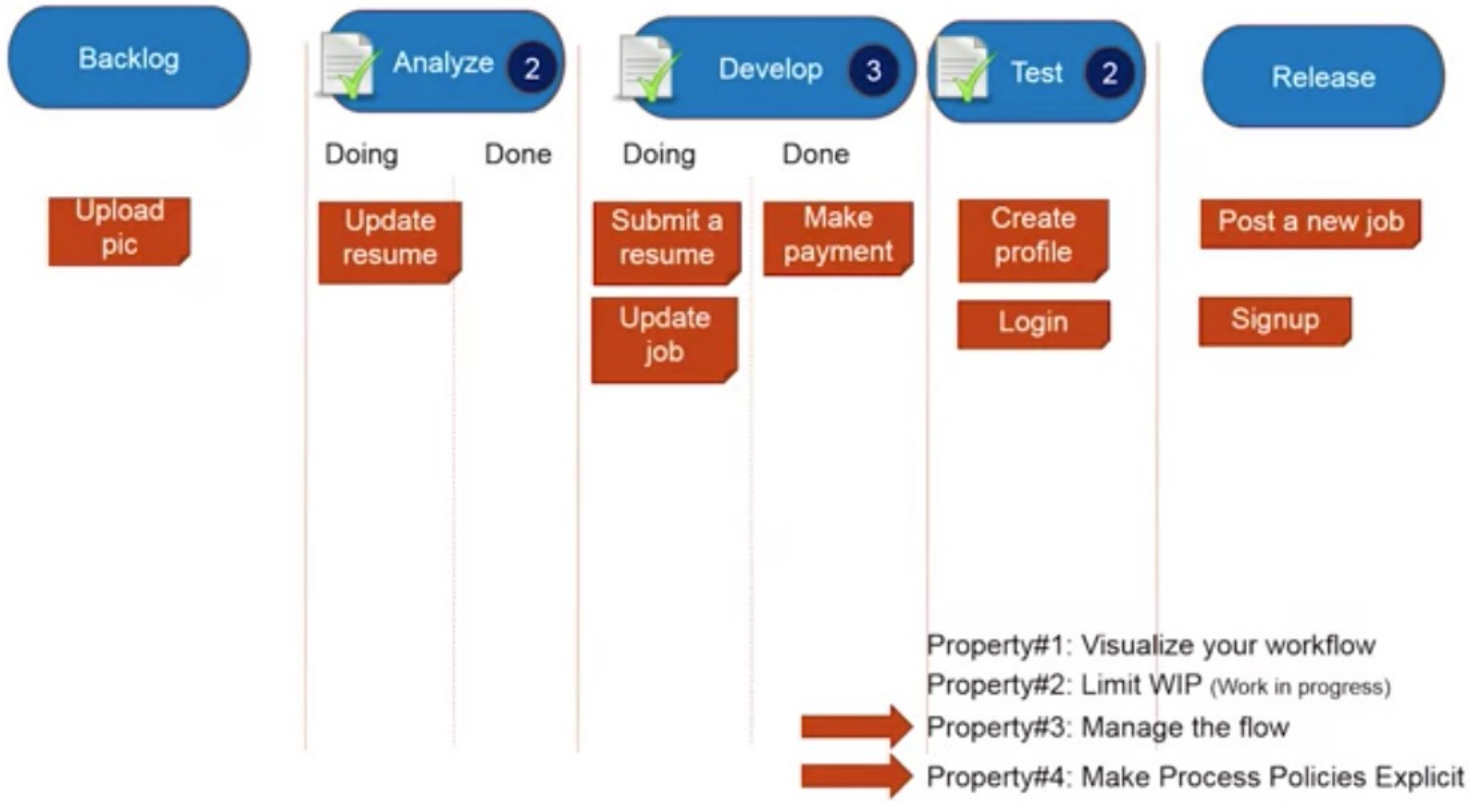
KANBAN


After few days…
Backlog
Analyze 2
Develop 3
Test 2
Release
Doing
Done
Doing
Done
Upload pic
Update resume
Update job
Submit a resume
Create profile
Signup
Make payment
Login
Post a new job
Property#1: Visualize your workflow
Property#2: Limit WIP (Work in progress)

KANBAN


Backlog
Analyze 2
Develop 3
Test 2
Release
Doing
Done
Doing
Done
Upload pic
Update resume
Update job
Submit a resume
Make payment
Login
Create profile
Signup
Post a new job
Property#1: Visualize your workflow
Property#2: Limit WIP (Work in progress)

KANBAN


Backlog
Analyze 2
Develop 3
Test 2
Release
Doing
Done
Doing
Done
Upload pic
Update resume
Submit a resume
Make payment
Create profile
Post a new job
Update job
Login
Signup
Property#1: Visualize your workflow
Property#2: Limit WIP (Work in progress)
Property#3: Manage the flow
Property#4: Make Process Policies Explicit

KANBAN PRINCIPLES

1. Start with what you do know
2. Agree to pursue incremental, evolutionary change
3. Respect the current process, roles, responsibilities & titles

KANBAN PROPERTIES

1. Visualize the workflow
2. Limit WIP (work in progress)
3. Manage flow
4. Make process policies explicit
5. Improve Collaboratively

AGILE AND LEAN

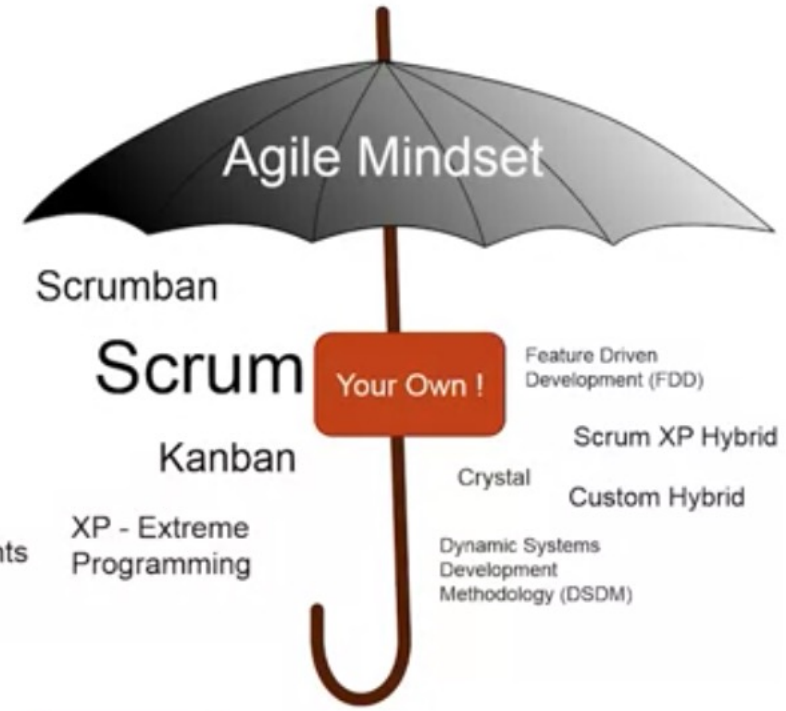
- Mindset not process/ model
- Incremental and Iterative
- Very short dev cycle

- Flexible to change increases chances of building the right product
- Speed to market

USE

- Where requirements may change
- Team using unproven technology

- May result in rework
- Requires close collaboration with clients and users



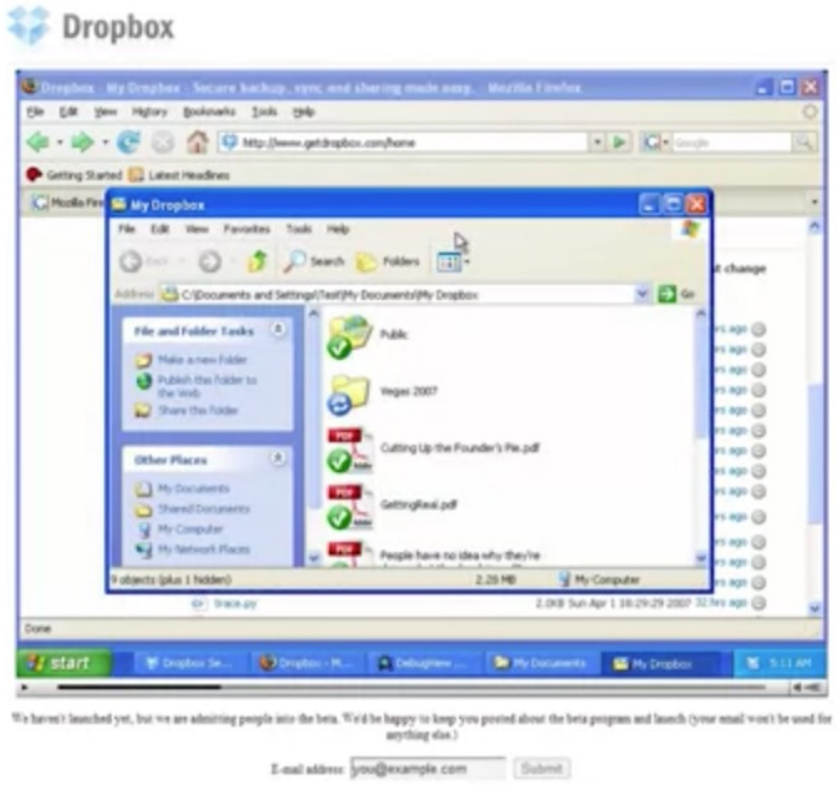
WHAT DOES IT LOOK LIKE?

We must learn what customers really want, not what they say they want or what we think they should want. We must discover whether we are on a path that will lead to growing a sustainable business." - Eric Ries

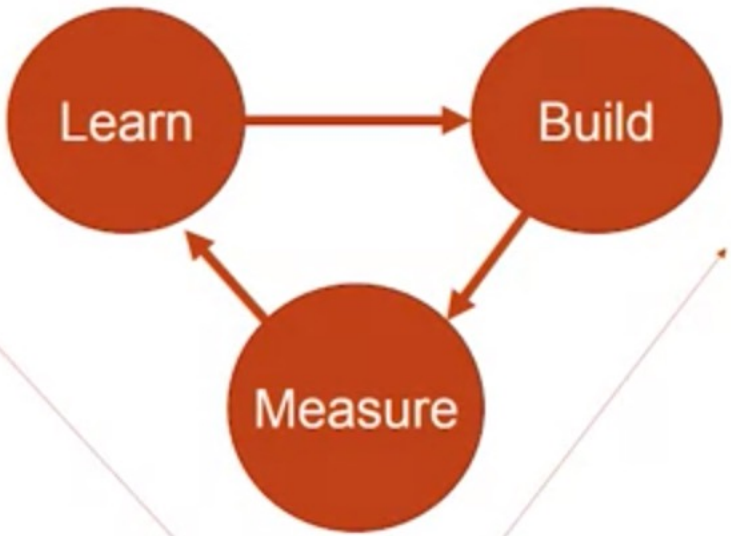
LEAN STARTUP

Focuses on understanding real user needs.

EXAMPLE


Dropbox
My Dropbox
File and Folder Tasks
Other Places
Public
Vegas 2007
Cutting Up the Founder's Pie.pdf
GettingReal.pdf
People have no idea why they're
start
E-mail address:
you@example.com
Submit

Assumption:
People would want the functionality to sync file on multiple platform



Experiment:
Create a video that will show how it will work.

Dropbox

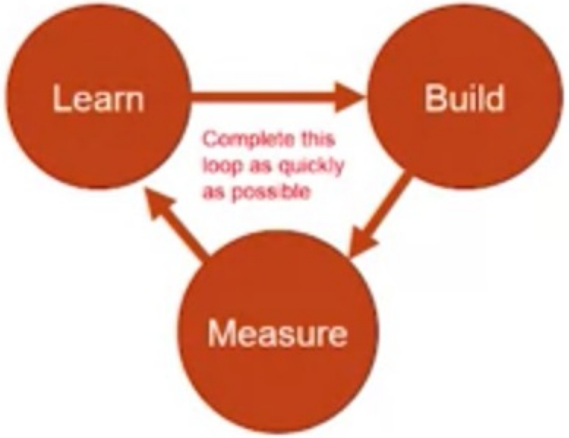
Metric: # of people signup to use the service

LEAN STARTUP

- Incremental and Iterative
- Very short dev cycle

USE

- Doubtful Business case / user need
- Lots of high probability risks



- Helps you learn faster and build the right product
- Speed to market

- May result in rework
- Requires you to experiment iteratively with clients and users