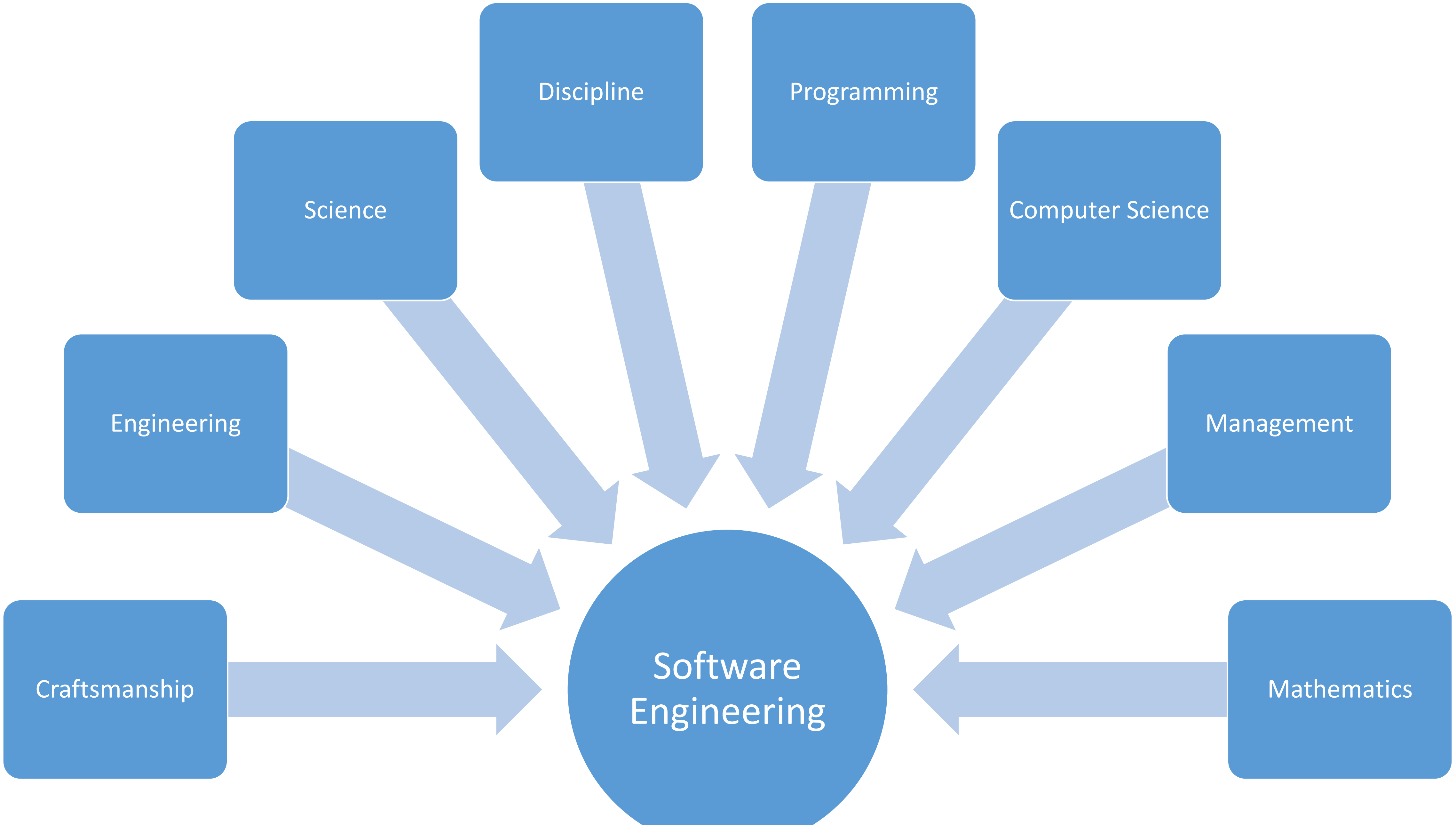
The Life of Humans
&
The Life of Software

What is Software Engineering anyway?

What is Software Engineering anyway?

- “A discipline that deals with the building of software systems which are so large that they are built by a team or teams of engineers.” [Ghezzi, Jazayeri, Mandrioli]
- “Multi-person construction of multi-version software.” [Parnas]
- “A discipline whose aim is the production of fault-free software, delivered on-time and within budget, that satisfies the user’s needs.” [Schach]
- “Interesting & Challenging” [Chimalakonda]
- “is a systematic approach to design of software systems with desired qualities under given constraints” [Chimalakonda]

What is Software Engineering anyway?



What is Software Engineering?

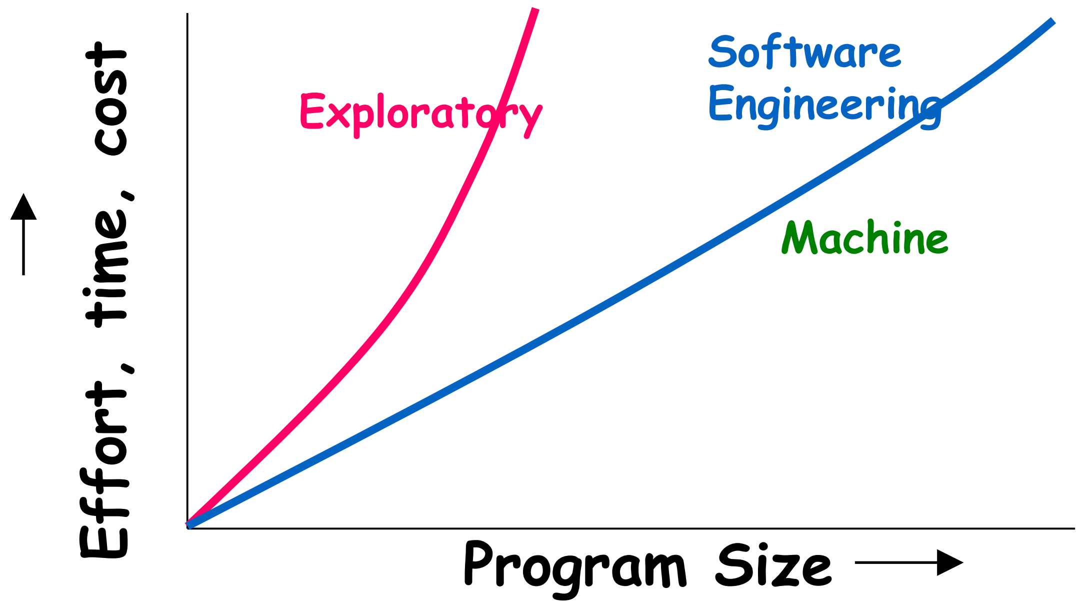
- Engineering approach to develop software.
  - Building Construction Analogy.
- Systematic collection of past experience:
  - Techniques,
  - Methodologies,
  - Guidelines.

Exploratory programming to Software Engineering

- The early programmers used an exploratory (also called build and fix) style.
    - In the build and fix (exploratory) style, normally a `dirty' program is quickly developed.
    - The different imperfections that are subsequently noticed are fixed.

What is Wrong with the Exploratory Style?

- Can successfully be used for very small programs only.



What is Wrong with the Exploratory Style? Contd…

- Besides the exponential growth of effort, cost, and time with problem size:
  - Exploratory style usually results in unmaintainable code.
  - It becomes very difficult to use the exploratory style in a team development environment.

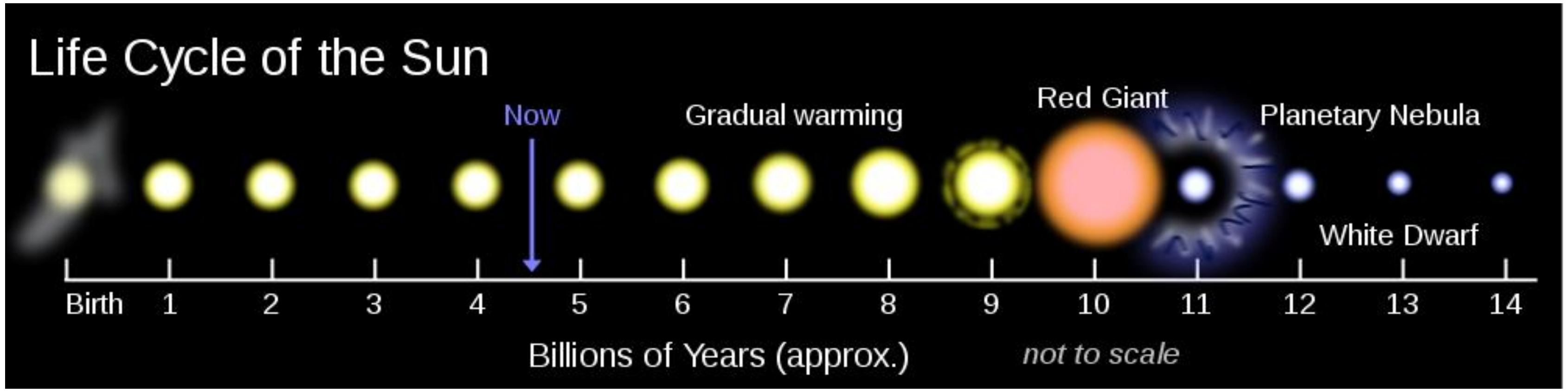
What is Wrong with the Exploratory Style? Contd…

- Why does the effort required to develop a product grow exponentially with product size?

  - Why does the approach completely break down when the product size becomes large?

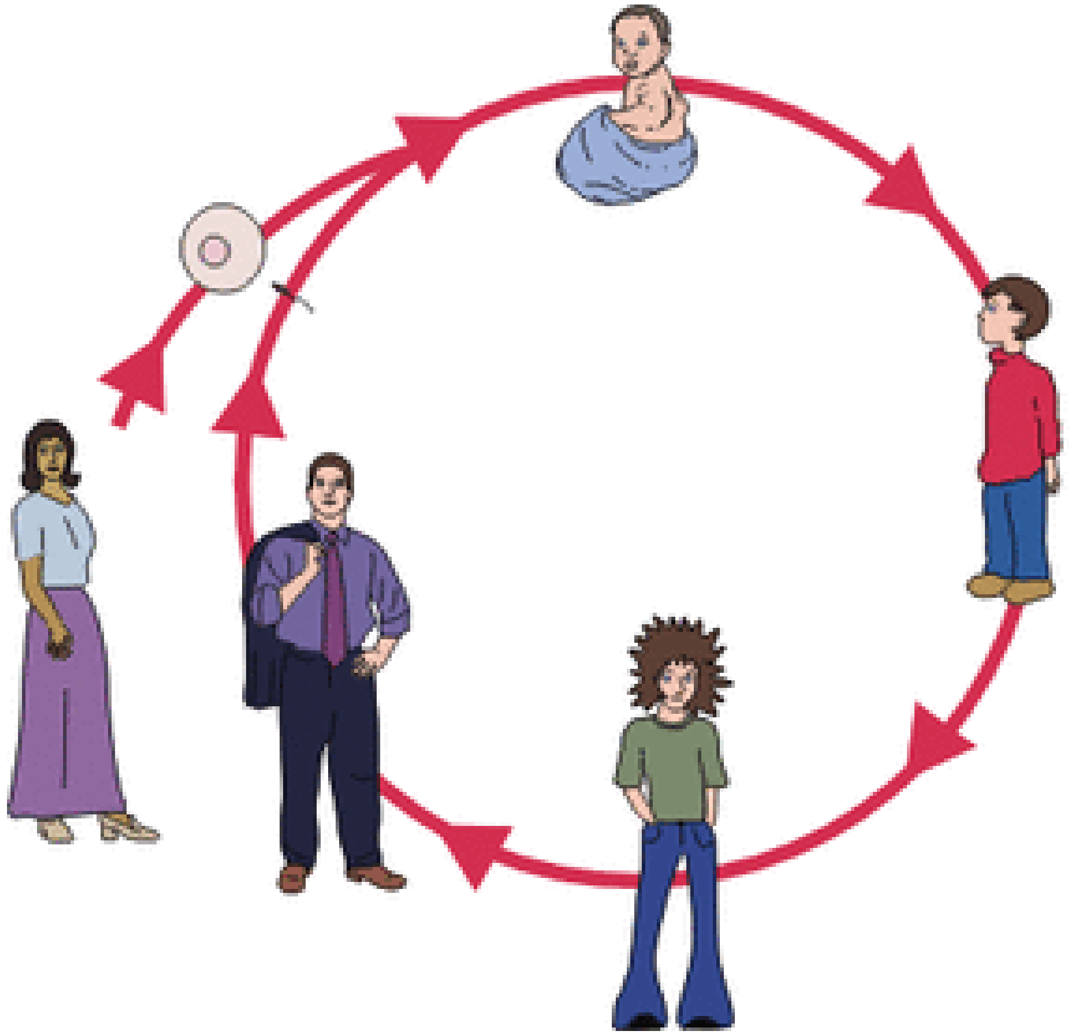
Don’t you think it is crazy?
We are talking of life for a non-living thing like *software?*

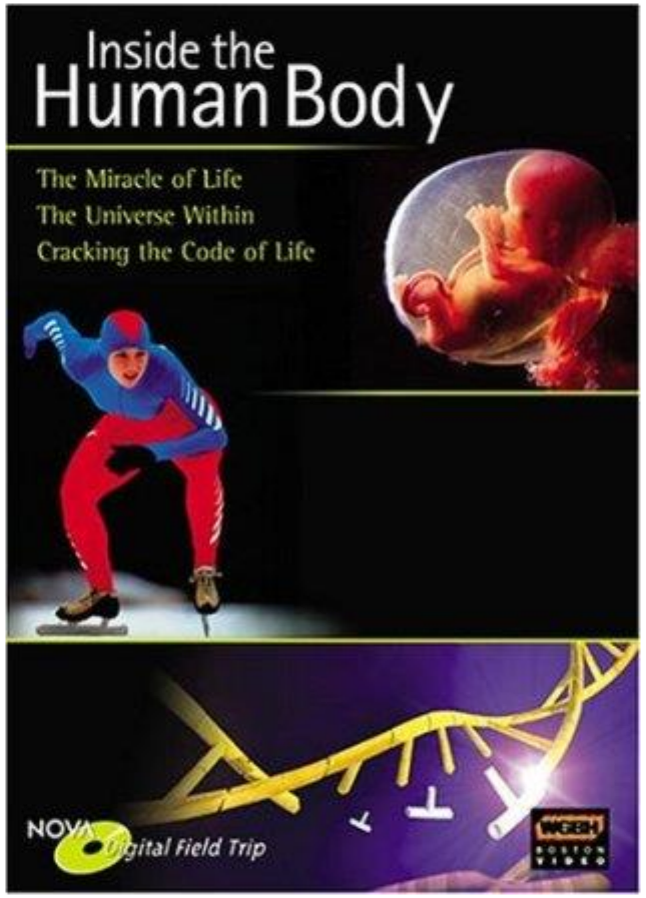
What do you do in Life? Everyday? Week? Month? Years?



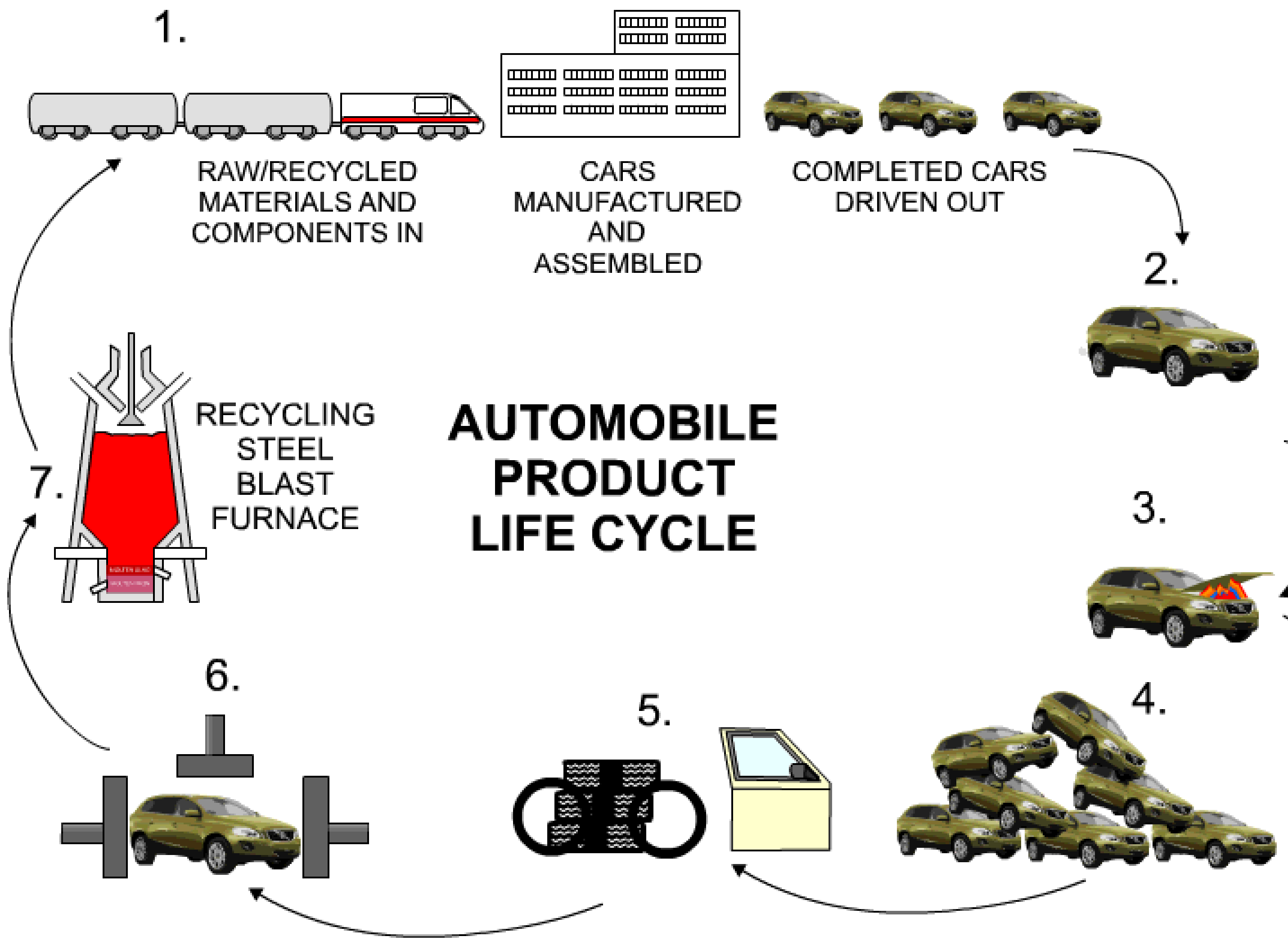
So, what is life cycle?

Life of Humans, Plants…





Now, life of man-made things


1.
RAW/RECYCLED MATERIALS AND COMPONENTS IN
CARS MANUFACTURED AND ASSEMBLED
COMPLETED CARS DRIVEN OUT
2.
3.
4.
5.
6.
7.
RECYCLING STEEL BLAST FURNACE
AUTOMOBILE PRODUCT LIFE CYCLE

Everything has a life and cycle

- Life cycle of Product [even *idea*]
- What happens during this evolution?
- What if you have to do it in a better way?

- Ideas, Thoughts….
- Concrete

What about software life cycle?

Influences – The Key Slide

- People
- Processes
- Tools
- Products

- Explicit Versus Implicit
- Tangible Versus Intangible
- Manageable Complexity Versus Unmanageable Complexity
- Changeable Environment Versus
- Unchangeable Environment
- No Major Changes Versus Major Changes

Does it always work?

People	Processes	Tools	Product
Good	Good	Good	Good
Good	Good	Good	Bad
Good	Good	Bad	Good
Good	Good	Bad	Bad
Good	Bad	Good	Good
Good	Bad	Good	Bad
Bad	Good	Good	Good
Bad	Good	Good	Bad
Good	Bad	Bad	Good
Good	Bad	Bad	Bad
Bad	Good	Bad	Good
Bad	Good	Bad	Bad
Bad	Bad	Good	Good
Bad	Bad	Good	Bad
Bad	Bad	Bad	Good
Bad	Bad	Bad	Bad

WHY LIFE CYCLE MODEL?

- A software project will never succeed if activities are not coordinated:
    - one engineer starts writing code,
    - another concentrates on writing the test document first,
    - yet another engineer first defines the file structure
    - another defines the I/O for his portion first
- Adherence can lead to accurate status reports
- Otherwise, it becomes very difficult to track the progress of the project
    - the project manager would have to depend on the guesses of the team members.

LIFE CYCLE MODEL

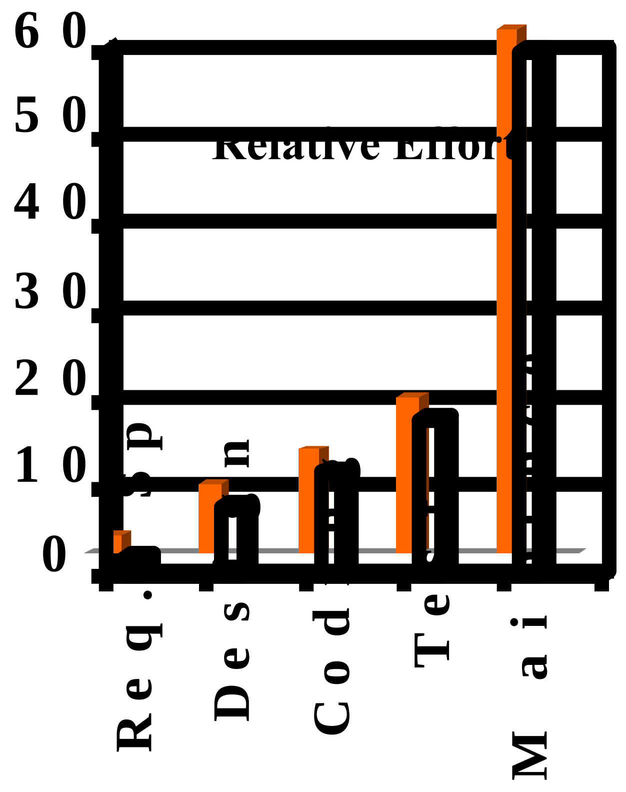
- A software life cycle model (or process model):
  - a descriptive and diagrammatic model of software life cycle:
  - identifies all the activities required for product development
  - establishes a precedence ordering among the different activities
  - divides life cycle into phases.

SOFTWARE DEVELOPMENT LIFE CYCLE

- Typical software life cycle or software process consists of following phases:
  - Feasibility study (involves business case)
  - Requirements analysis and specification,
  - Design
  - Coding
  - Testing
  - Maintenance

RELATIVE EFFORT FOR PHASES

- Phases between feasibility study and testing
  - known as development phases.
- Among all life cycle phases
  - maintenance phase consumes maximum effort.



FEASIBILITY STUDY

- Main aim of feasibility study: determine whether developing the product
  - financially worthwhile
  - technically feasible.

- First roughly understand what the customer wants:
  - Inputs
  - Processing
  - Outputs
  - various constraints on the behaviour of the system

ACTIVITIES DURING FEASIBILITY STUDY

- Work out an overall understanding of the problem
- Formulate different solution strategies
- Examine alternate solution strategies in terms of:
    - resources required
    - cost of development
    - development time
- Perform a cost/benefit analysis:
    - you may determine that none of the solutions is feasible due to high cost, resource constraints, technical reasons.

REQUIREMENTS ANALYSIS AND SPECIFICATION

- <u>Aim of this phase:</u>
  - understand the <u>exact requirements</u> of the customer,
  - document them properly.

- Consists of two distinct activities:
  - requirements gathering and analysis
  - requirements specification.

GOALS OF REQUIREMENTS ANALYSIS

- Collect all related data from the customer:
  - analyze the collected data to clearly understand what the customer wants,
  - ensure correctness, consistency and unambiguity.

REQUIREMENTS GATHERING

- Gathering relevant data:
  - usually collected from the end-users through interviews and discussions.
  - For example, for a business accounting software:
    - interview all the accountants of the organization to find out their requirements.

REQUIREMENTS ANALYSIS (CONT.)

- The data you initially collect from the users:
  - would usually contain several contradictions and ambiguities:
  - each user typically has only a partial and incomplete view of the system.

REQUIREMENTS ANALYSIS (CONT.)

- Ambiguities and contradictions:
  - must be identified
  - resolved by discussions with the customers.
- Next, requirements are organized:
  - into a **Software Requirements Specification (SRS)** document.

DESIGN

- Design phase transforms requirements specification:
  - into a form suitable for implementation in some programming language.

DESIGN

- <u>High-level design:</u>
  - decompose the system into ***modules***,
  - represent invocation relationships among the modules.

- <u>Detailed design:</u>
  - different modules designed in greater detail:
    - data structures and algorithms for each module are designed.

IMPLEMENTATION

- During the implementation phase:
  - each module of the design is coded,
  - each module is unit tested
    - tested independently as a stand alone unit, and debugged

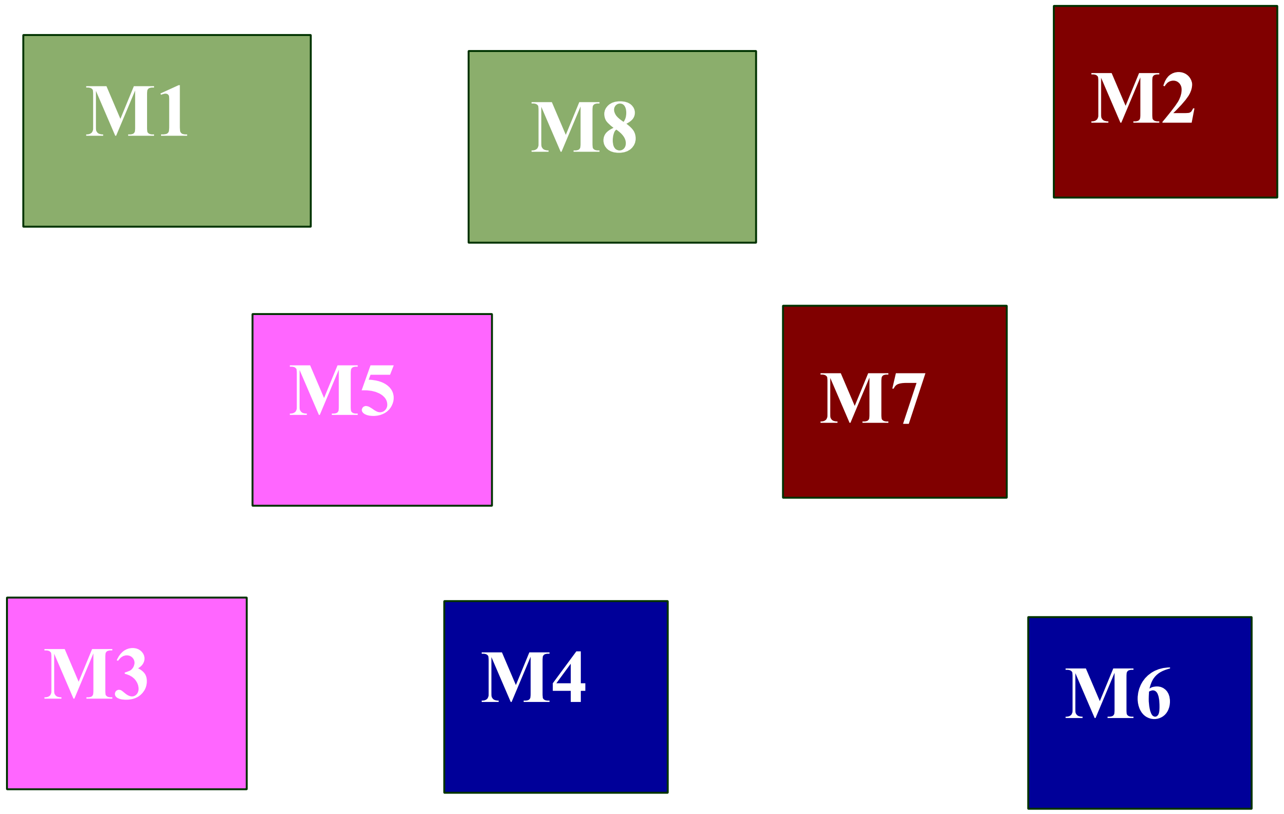
IMPLEMENTATION (CONT.)

- The purpose of unit testing:
  - test if individual modules work correctly.
- The end product of implementation phase:
  - a set of program modules that have been tested individually.

INTEGRATION AND SYSTEM TESTING

- Different modules are integrated in a planned manner:
  - modules are almost never integrated in one shot.
  - Normally integration is carried out through a number of steps.
- During each integration step,
  - the partially integrated system is tested.

INTEGRATION AND SYSTEM TESTING



SYSTEM TESTING

- After all the modules have been successfully integrated and tested:
  - system testing is carried out.
- <u>Goal of system testing:</u>
  - ensure that the developed system functions according to its requirements as specified in the SRS document.

MAINTENANCE

- Maintenance of any software product:
  - requires much more effort than the effort to develop the product itself.
  - development effort to maintenance effort is typically 40:60.

MAINTENANCE (CONT.)

- Preventive maintenance
  - Making appropriate changes to prevent the occurrence of errors
- Corrective maintenance
  - Correct errors which were not discovered during the product development phases
- Perfective maintenance
  - Improve implementation of the system
  - enhance functionalities of the system
- Adaptive maintenance
  - Port software to a new environment

SUMMARY

- A software life cycle model (or process model):
  - a descriptive and diagrammatic model of software life cycle
  - identifies all the activities required for product development,
  - establishes a precedence ordering among the different activities
  - divides life cycle into phases.
- A fundamental necessity while developing any large software product:
  - Adoption of a software development life cycle model (software process model).

Project Proposal Scope and activities

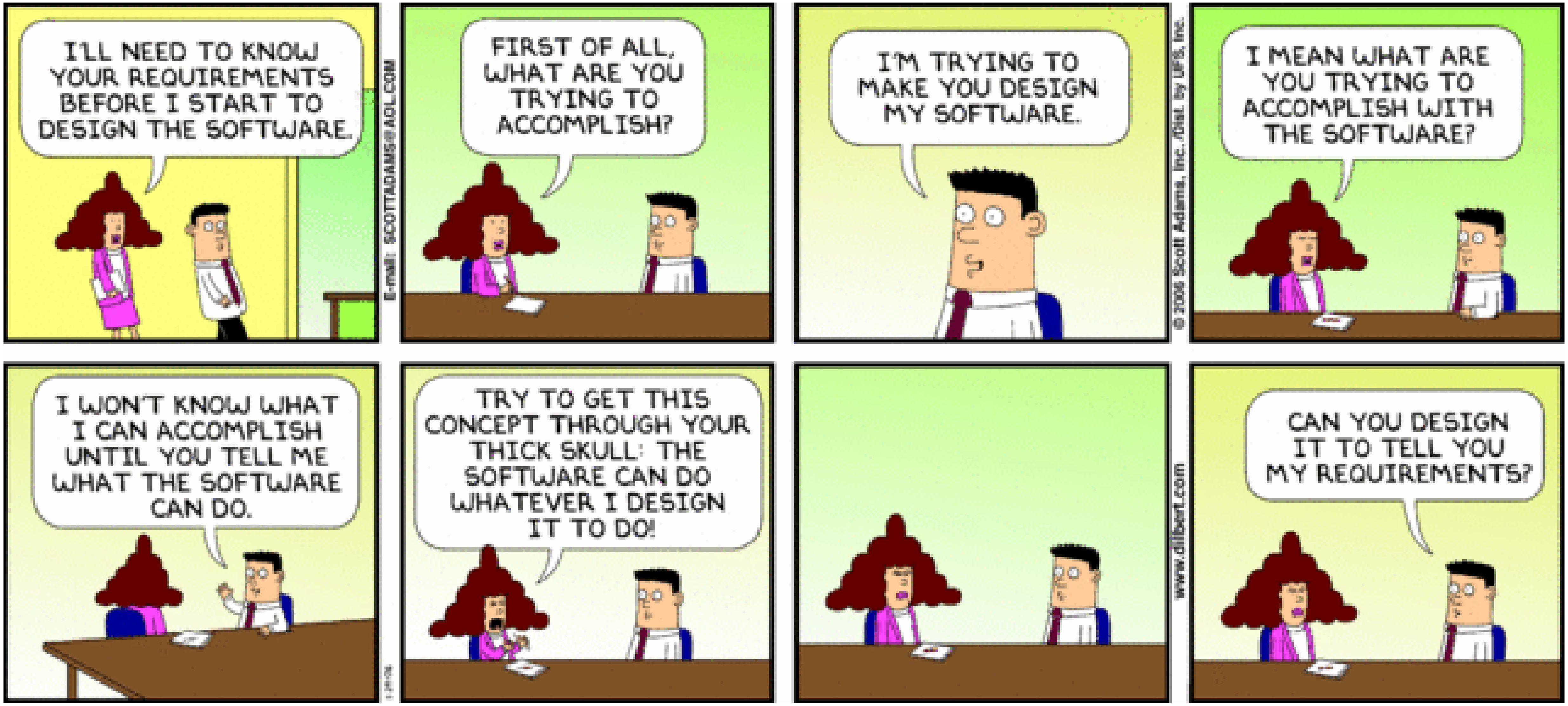
- Ill-defined requirements
- Customers
- Time-pressure
- Teamwork
- Different team roles
- Control over design
- …

Project Development

- Proposal
- Requirements
- Design
- Implementation
- Testing, validation, verification
- Documentation
- Customer exposure
- Final deliverable

Learnings from past experience

- A slow start
- Insufficient team meeting time
- Choosing project software solely because you want to learn it
- Ignoring the importance of understanding the domain
- Too much time making non-critical decisions
- Too much time making critical decisions
- “Super-programmers” who try to take over and make it a “mere matter of programming”
- Too much/too little time getting tools to work
- Too much/too little focus on documentation
- Isolating or marginalizing one or more team members
- Assuming nothing will go wrong
- Overly high expectations about what is achievable
- Nothing works unless everything works

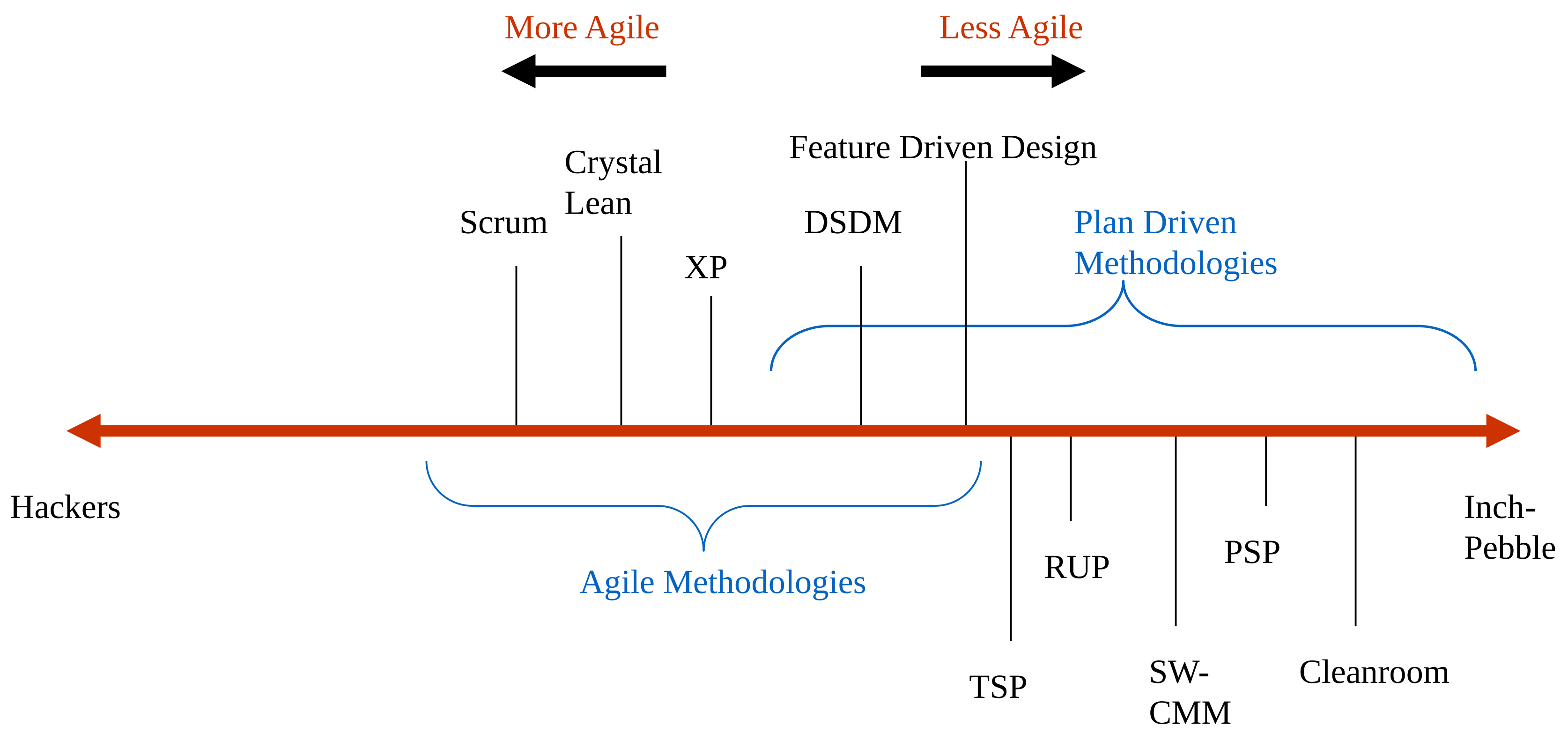

I'LL NEED TO KNOW YOUR REQUIREMENTS BEFORE I START TO DESIGN THE SOFTWARE.
FIRST OF ALL, WHAT ARE YOU TRYING TO ACCOMPLISH?
I'M TRYING TO MAKE YOU DESIGN MY SOFTWARE.
I MEAN WHAT ARE YOU TRYING TO ACCOMPLISH WITH THE SOFTWARE?
I WON'T KNOW WHAT I CAN ACCOMPLISH UNTIL YOU TELL ME WHAT THE SOFTWARE CAN DO.
TRY TO GET THIS CONCEPT THROUGH YOUR THICK SKULL: THE SOFTWARE CAN DO WHATEVER I DESIGN IT TO DO!
CAN YOU DESIGN IT TO TELL YOU MY REQUIREMENTS?
E-mail: SCOTTADAMS@AOL.COM
© 2006 Scott Adams, Inc./Dist. by UFS, Inc.
www.dilbert.com

PROCESS MODELS

Many life cycle models have been proposed

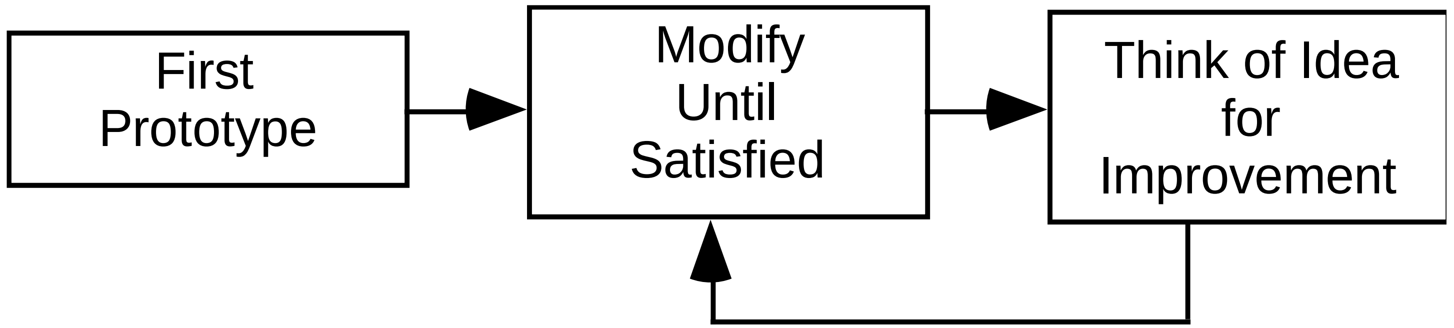
- Traditional Models (plan-driven)
  - Classical waterfall model
  - Iterative waterfall
  - Evolutionary
  - Prototyping
  - Spiral model
  - Rational Unified Process (RUP)
- Agile Models
  - eXtreme Programming (XP)
  - Scrum
  - Crystal
  - Feature-Driven Development (FDD)

The Process Methodology Spectrum



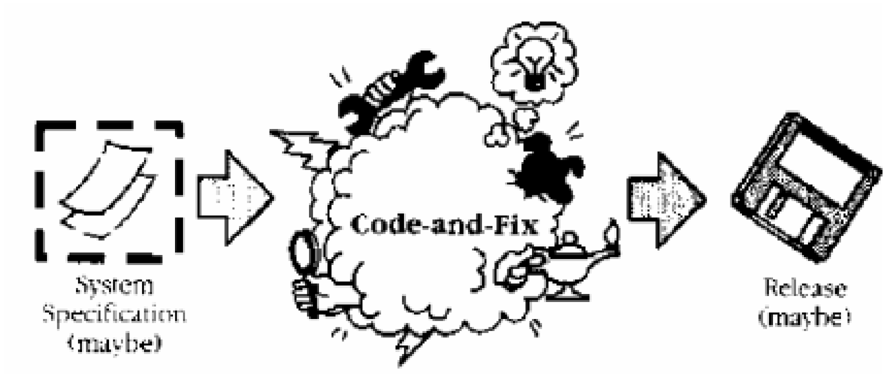
from “Balancing Agility & Discipline” (Boehm & Turner)

The opportunistic approach



- OK for small, informal projects
- Inappropriate for professional environments/complex software where on-time delivery and high quality are expected

Ad hoc lifecycle



- "Go for it!"
- Advantages
    - Very easy to learn and to apply
    - Might work in some small and short-lived projects

Disadvantages?

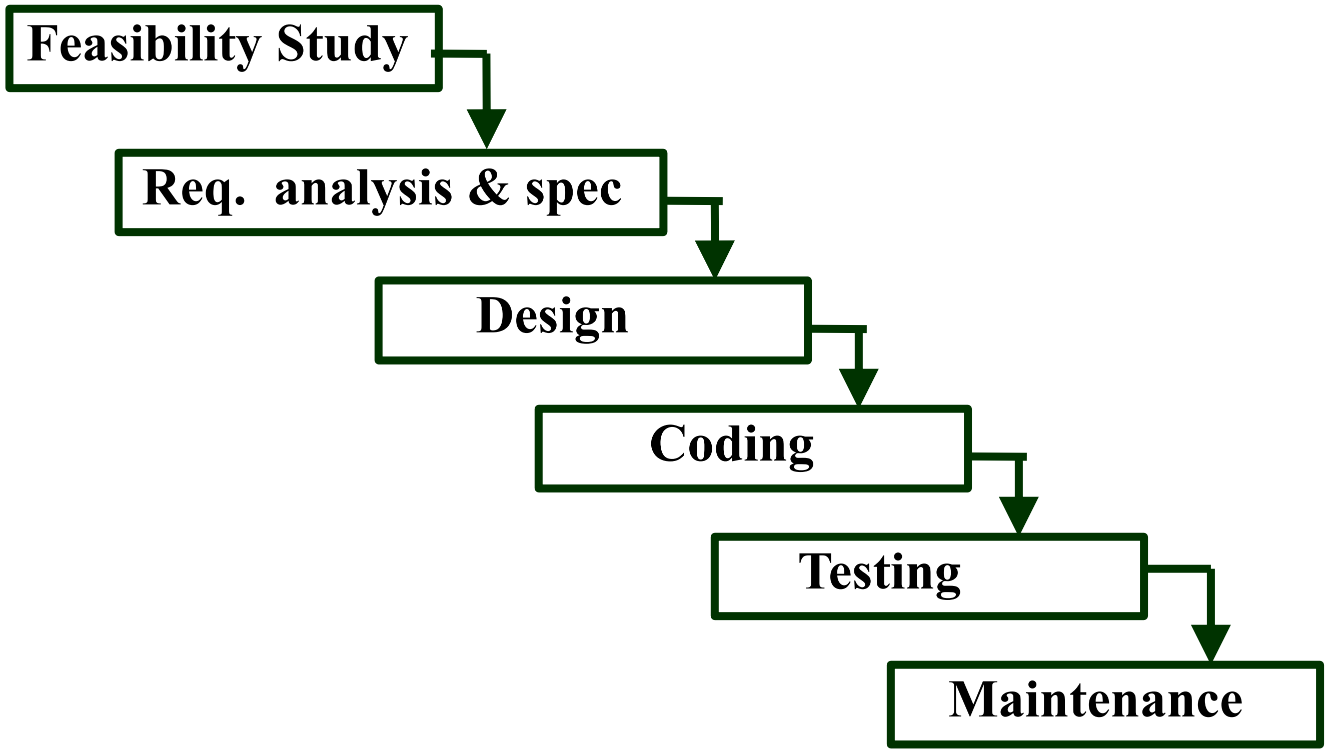
Lack of Scalability:
The lack of initial planning may result in architectural bottlenecks which require reworking.

Technical Debt:
Quick fixes or Workarounds that may not be optimal in the long term.

Lack of Reusability: Code is often written without keeping reusability and mentainablility in mind. which causes issues.

Higher Costs in Long Run

CLASSICAL WATERFALL MODEL



CLASSICAL WATERFALL MODEL (CONT.)

- The guidelines and methodologies of an organization:
  - called the organization's software development methodology.
- Software development organizations:
  - expect fresh engineers to master the organization's software development methodology.

Problems with Classical Waterfall Model

- Classical waterfall model is idealistic:
  - assumes that no defect is introduced during any ~~development activity.~~ phase of life-cycle model
  - in practice:
    - defects do get introduced in almost every phase of the life cycle.
- Defects usually get detected much later in the life cycle:
  - For example, a design defect might go unnoticed till the coding or testing phase

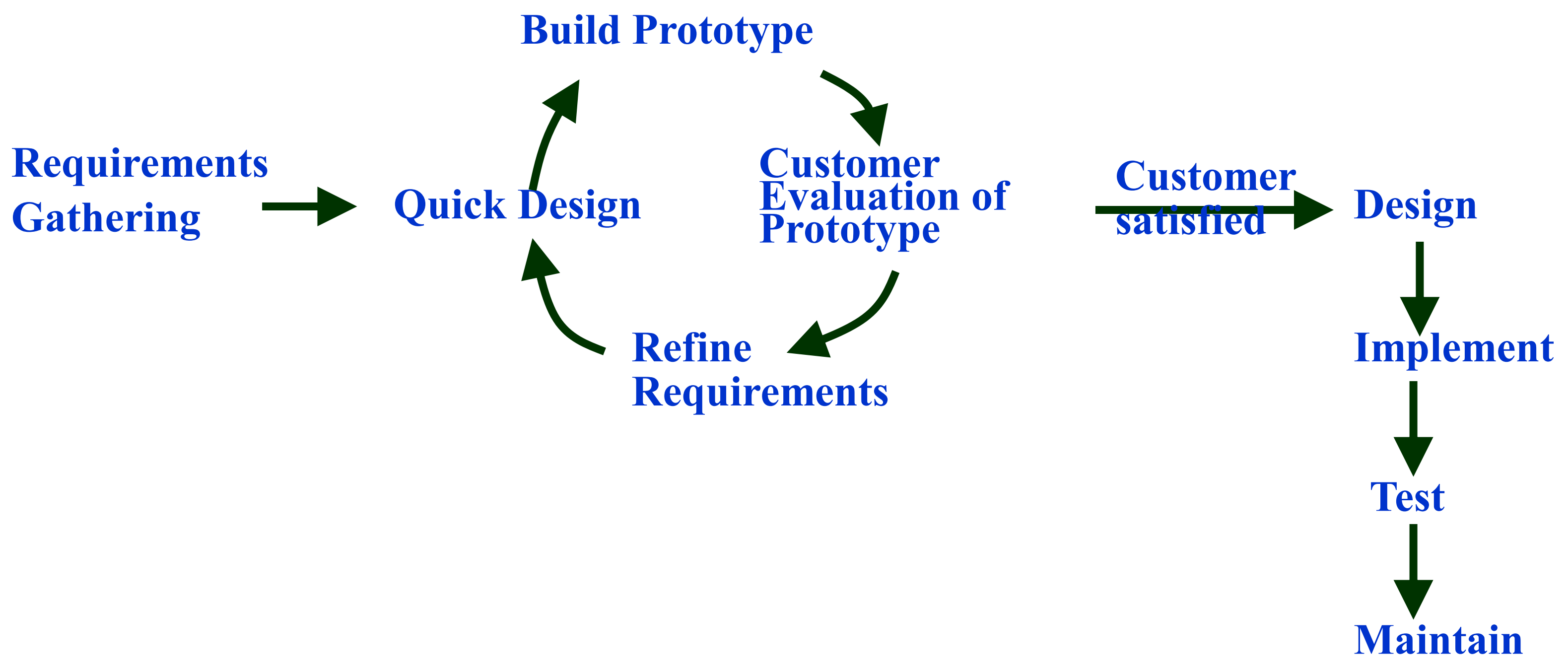
PROTOTYPING MODEL

- Before starting actual development,
  - a working prototype of the system should first be built.

- A prototype is a toy implementation of a system:
  - limited functional capabilities,
  - low reliability,
  - inefficient performance.

WHY PROTOTYPE?

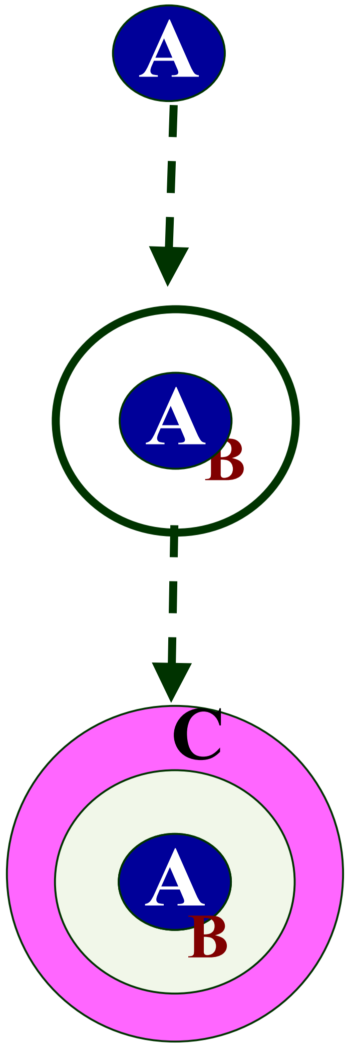
PROTOTYPING MODEL (CONT.)

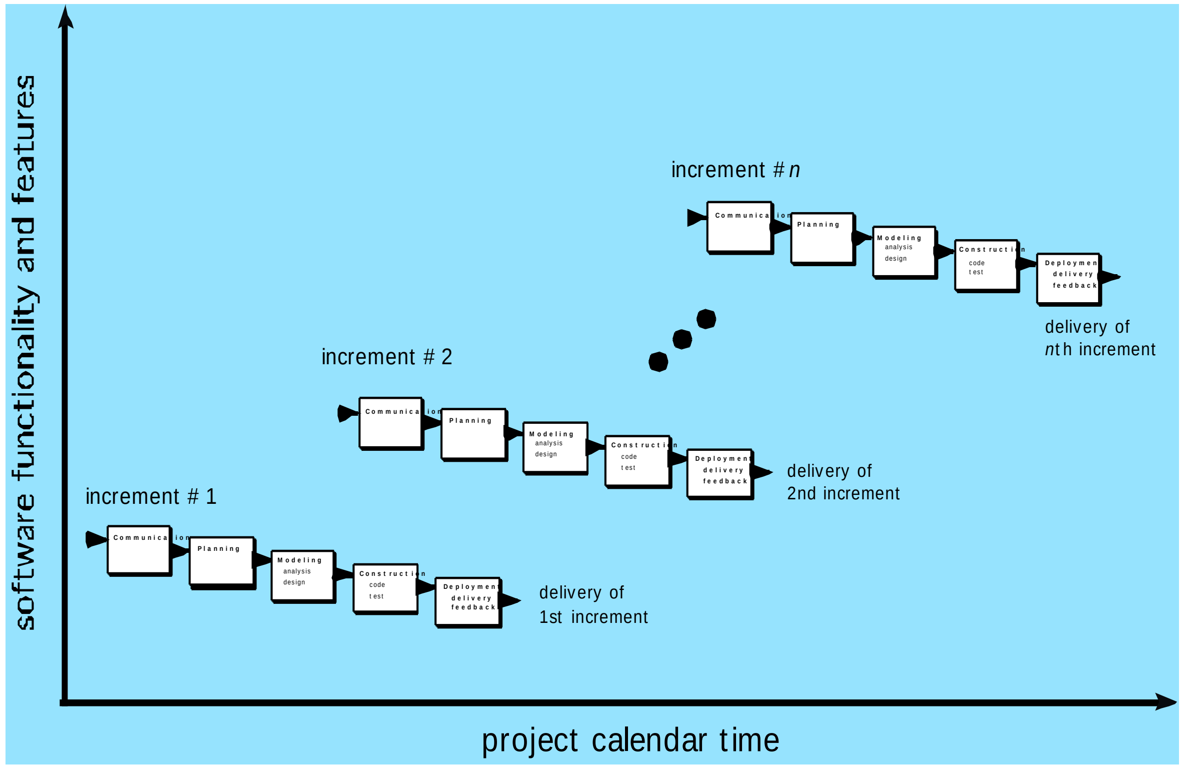


EVOLUTIONARY MODEL

- Evolutionary model (aka successive versions or incremental model):
  - The system is broken down into several modules which can be incrementally implemented and delivered.
- First develop the core modules of the system.
- The initial product skeleton is refined into increasing levels of capability:
  - by adding new functionalities in successive versions.

INCREMENTAL MODEL





ADVANTAGES OF EVOLUTIONARY MODEL

- Users get a chance to experiment with a partially developed system:
  - much before the full working version is released,
- Helps finding exact user requirements:
  - much before fully working system is developed.
- Core modules get tested thoroughly:
  - reduces chances of errors in final product.

DISADVANTAGES OF EVOLUTIONARY MODEL

- Often, difficult to subdivide problems into functional units:
  - which can be incrementally implemented and delivered.
  - evolutionary model is useful for very large problems,
    - where it is easier to find modules for incremental implementation.

EVOLUTIONARY MODEL WITH ITERATION (Iterative Incremental Model)

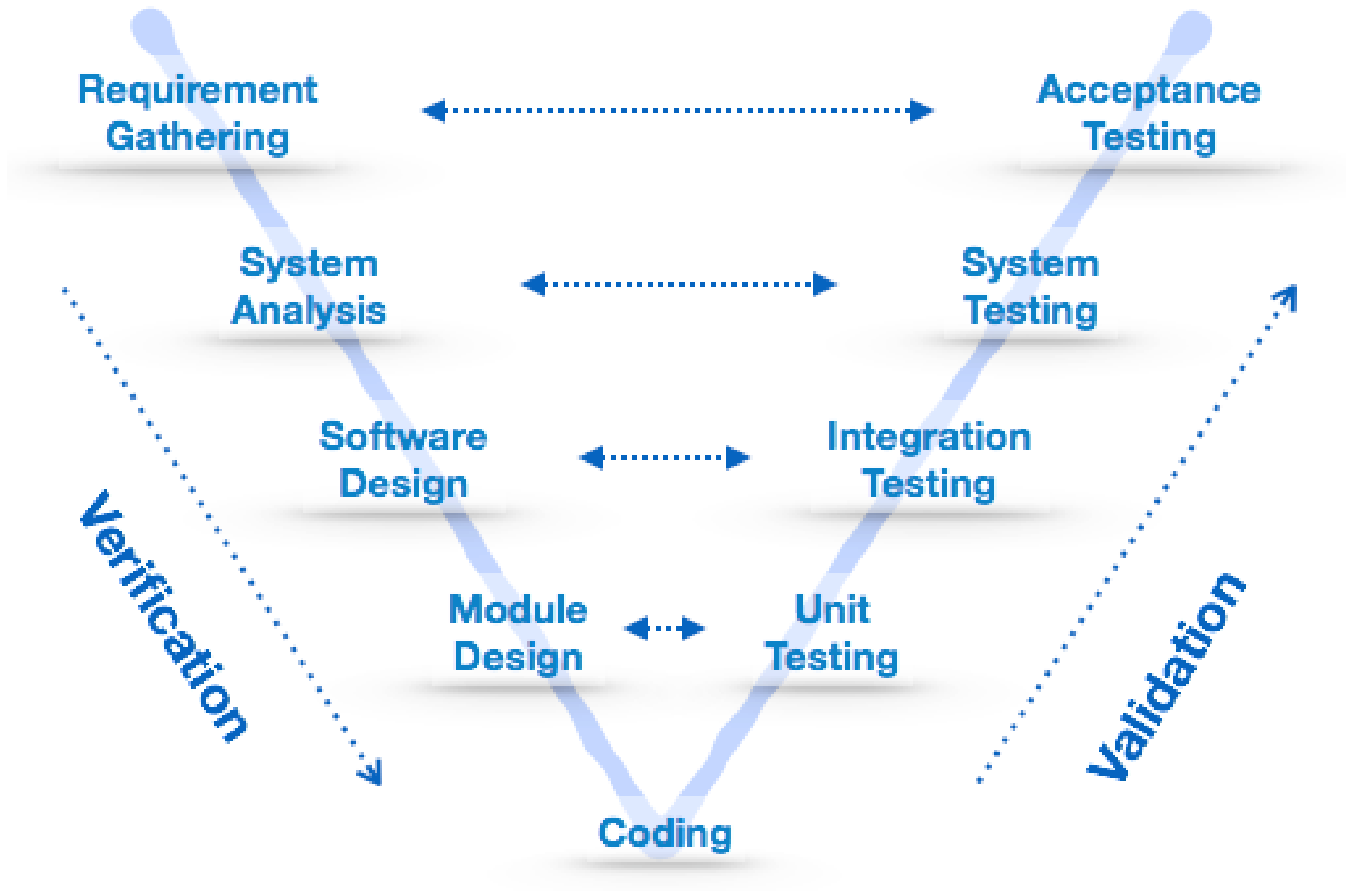
- Many organizations use a combination of iterative and incremental development:
  - a new release may include new functionality
  - existing functionality from the current release may also have been modified.

EVOLUTIONARY MODEL WITH ITERATION

- Several advantages:
  - Training can start on an earlier release
    - customer feedback taken into account
  - Markets can be created:
    - for functionality that has never been offered.
  - Frequent releases allow developers to fix unanticipated problems quickly.

V-Model

Verification -
Are we building the product right?

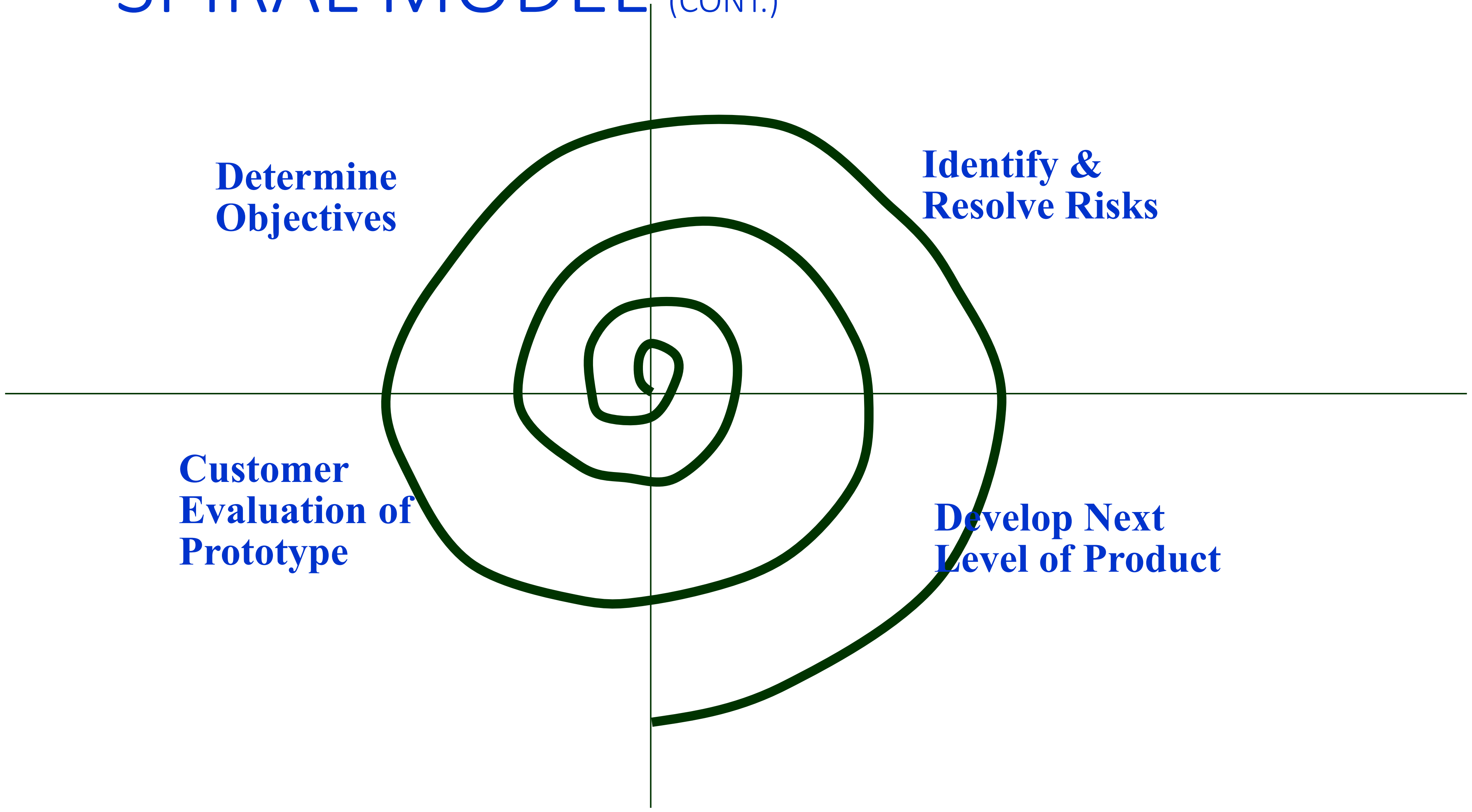


Validation -
Are we building the right product?

SPIRAL MODEL

- Proposed by Boehm in 1988.
- Each loop of the spiral represents a phase of the software process:
  - the innermost loop might be concerned with system feasibility,
  - the next loop with system requirements definition,
  - the next one with system design, and so on.
- There are no fixed phases in this model, the phases shown in the figure are just examples.

SPIRAL MODEL (CONT.)


Determine Objectives
Identify & Resolve Risks
Customer Evaluation of Prototype
Develop Next Level of Product

OBJECTIVE SETTING (FIRST QUADRANT)

- Identify objectives of the phase,
- Examine the risks associated with these objectives.
  - Risk:
    - any adverse circumstance that might hamper successful completion of a software project.
- Find alternate solutions possible.

RISK ASSESSMENT AND REDUCTION (SECOND QUADRANT)

- For each identified project risk,
  - a detailed analysis is carried out.
- Steps are taken to reduce the risk.
- For example, if there is a risk that the requirements are inappropriate:
  - a prototype system may be developed.

SPIRAL MODEL (CONT.)

- Development and Validation (Third quadrant):
  - develop and validate the next level of the product.
- Review and Planning (Fourth quadrant):
  - review the results achieved so far with the customer and plan the next iteration around the spiral.
- With each iteration around the spiral:
  - progressively more complete version of the software gets built.

SPIRAL MODEL AS A META MODEL

Each Quadrant represents a phase in Soft Life cycle.
One Succesful completion of loop marks completes an iteration fo waterfall model

- Subsumes all discussed models:
  - a single loop spiral represents waterfall model.
  - uses an evolutionary approach --
    - iterations through the spiral are evolutionary levels.
  - enables understanding and reacting to risks during each iteration along the spiral.
  - uses:
    - prototyping as a risk reduction mechanism
    - retains the step-wise approach of the waterfall model.

COMPARISON OF DIFFERENT LIFE CYCLE MODELS

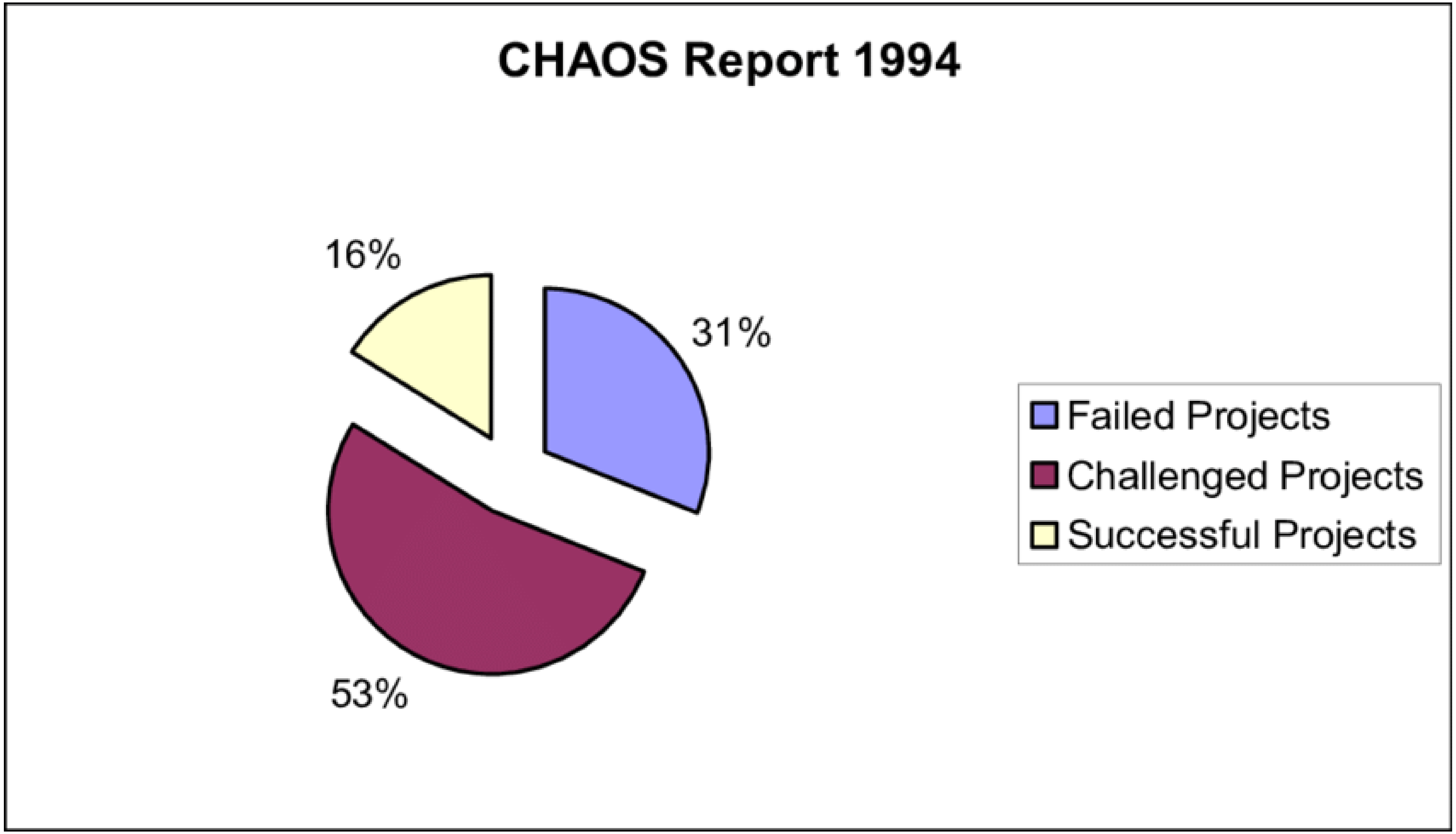
- Iterative waterfall model
  - most widely used model.
  - But, suitable only for well-understood problems.
- Prototype model is suitable for projects not well understood:
  - user requirements
  - technical aspects

COMPARISON OF DIFFERENT LIFE CYCLE MODELS (CONT.)

- Evolutionary model is suitable for large problems:
  - can be decomposed into a set of modules that can be incrementally implemented,
  - incremental delivery of the system is acceptable to the customer.
- The spiral model:
  - suitable for development of technically challenging software products that are subject to several kinds of risks.

Do software projects succeed?

CHAOS Report 1994

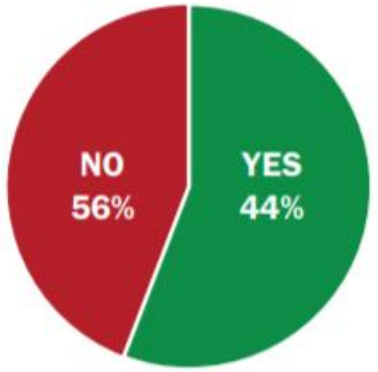


CHAOS REPORT 2015

The *CHAOS Report 2015* is a model for future *CHAOS Reports*. There have only been two previous *CHAOS Reports*, the original in 1994 and the 21st edition of 2014. This new type of *CHAOS Report* focuses on presenting the data in different forms with many charts. Most of the charts come from the new CHAOS database from the fiscal years 2011 to 2015. The CHAOS fiscal year starts March 1 and runs until the end of February. A few of the charts are from the new SURF database to highlight certain information. The purpose of this report is to present the data in the purest form without much analysis and little thought leadership. Analysis and thought leadership are offered in the *CHAOS Manifesto* series of reports.

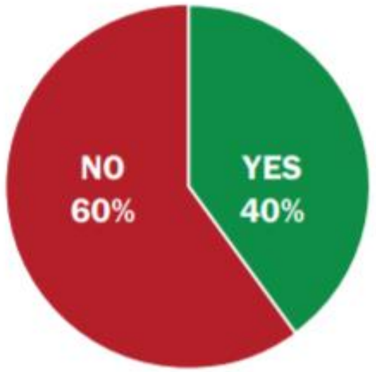
Another major change is how we define success. We have multiple definitions, including our newest. We coded the new CHAOS database with six individual attributes of success: OnTime, OnBudget, OnTarget, OnGoal, Value, and Satisfaction. Our Traditional definition is OnTime, OnBudget, and OnTarget. This means the project was resolved within a reasonable estimated time, stayed within budget, and contained a good number of the estimated features and functions. Our new Modern definition is OnTime, OnBudget, with a satisfactory result. This means the project was resolved within a reasonable estimated time, stayed within budget, and delivered customer and user satisfaction regardless of the original scope. We have the flexibility to present the results for one to six of these attributes in any combination.

ONBUDGET



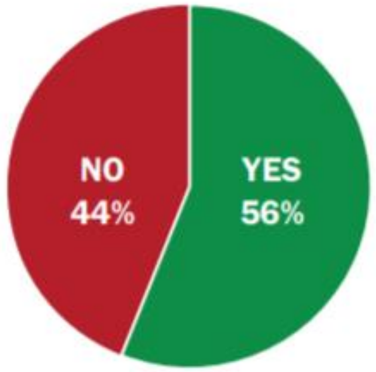
The percentage of projects that were

ONTIME

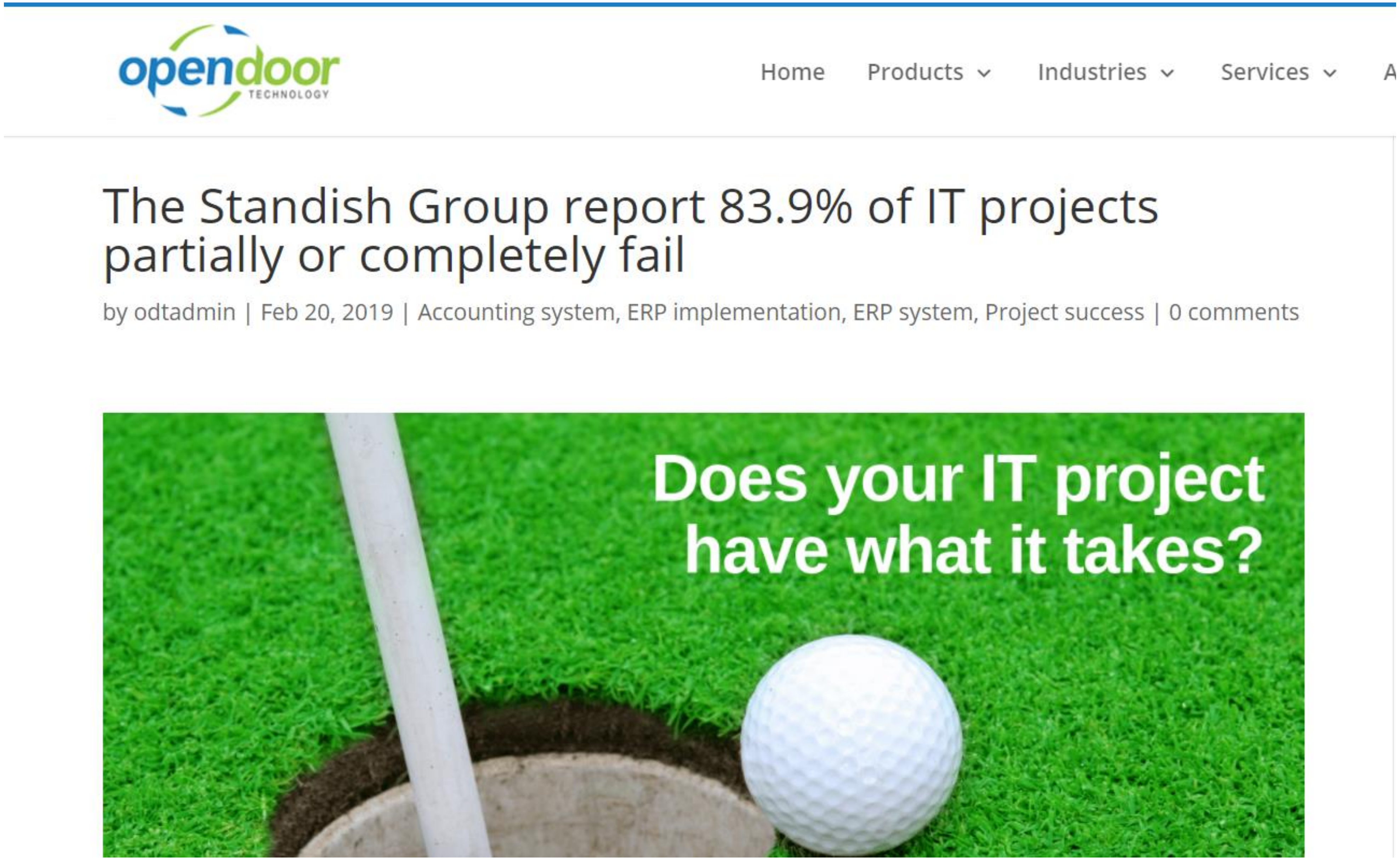


The percentage of projects that were

ONTARGET



The percentage of projects that were



The Standish Group report 83.9% of IT projects partially or completely fail

by odtadmin | Feb 20, 2019 | Accounting system, ERP implementation, ERP system, Project success | 0 comments

Agile processes

Agile Manifesto

Individuals and interactions	over	Process and tools
Working software	over	Comprehensive documentation
Customer collaboration	over	Contract negotiation
Responding to change	over	Following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Source: www.agilemanifesto.org

Some Agile Methods

- ASD - Adaptive Software Development
- Crystal
- FDD - Feature Driven Development
- DSDM - Dynamic Systems Development Method
- Lean Software Development
- Scrum
- XP - eXtreme Programming

Four Values

- Simplicity
  - create the simplest thing that could work
- Communication
  - face-to-face, not document-to-face
- Feedback
  - lots of tests
- Aggressiveness

Four Basic Activities

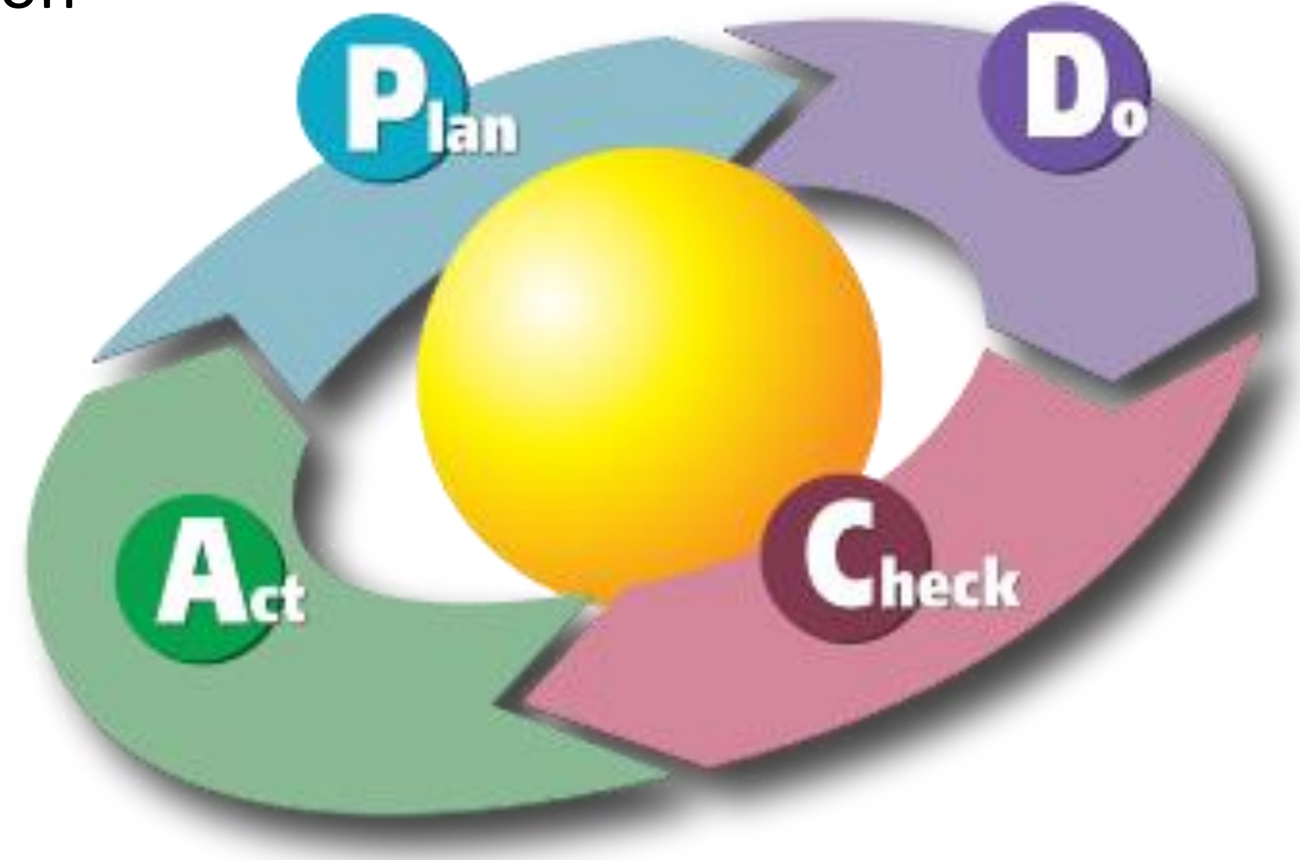
- Coding
  - cannot do without it
- Testing
  - if it cannot be tested it doesn't exist
- Listening
  - to those with domain knowledge
- Designing
  - to keep the system from decaying

Twelve Practices

1. The Planning Game
2. Small releases
3. Metaphor
4. Simple design
5. Testing
6. Refactoring
7. Pair programming
8. Collective ownership
9. Continuous integration
10. 40-hour week
11. On-site customer
12. Coding standards

Deming’s Philosophy

- Process-Product Co-relation

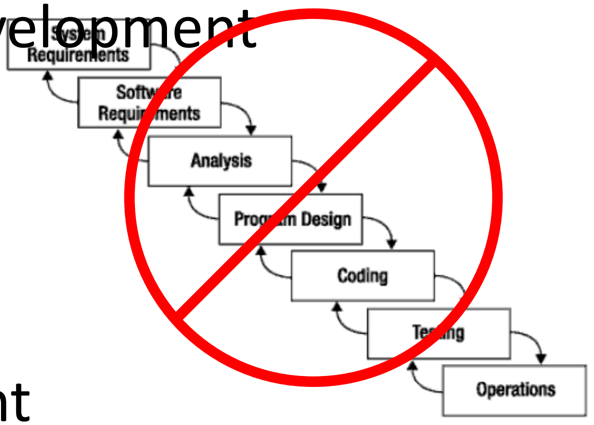


What is Scrum?

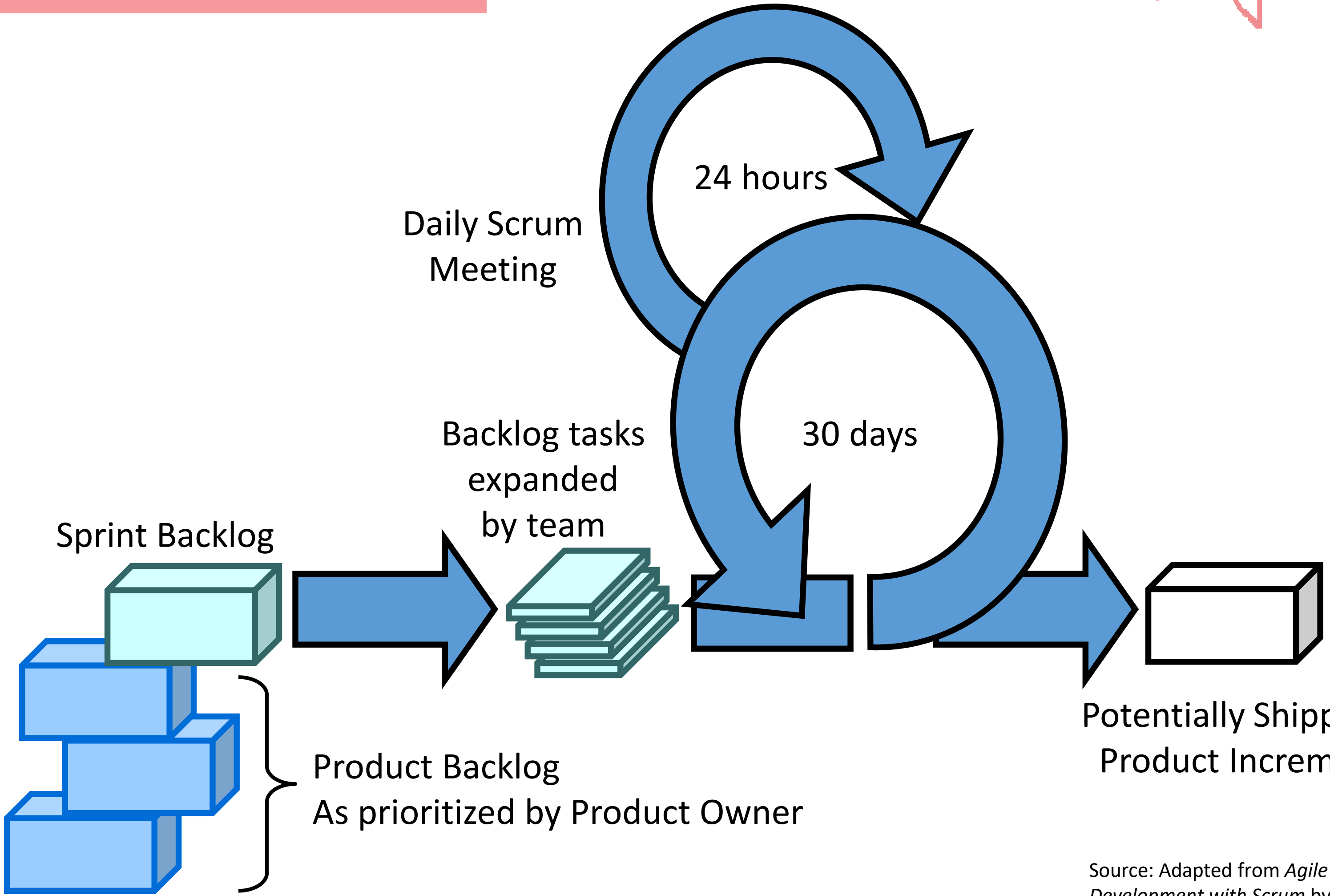
It's about common sense

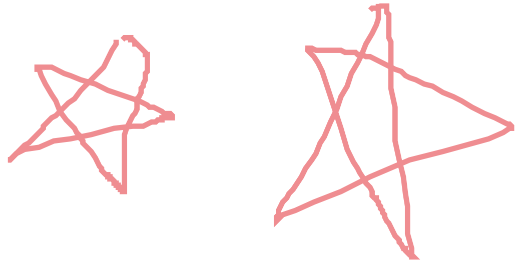
- **Scrum**:
  - Is an agile, lightweight process
  - Can manage and control software and product development
  - Uses iterative, incremental practices
  - Has a simple implementation
  - Increases productivity
  - Reduces time to benefits
  - Embraces adaptive, empirical systems development
  - Is not restricted to software development projects

  - Embraces the opposite of the waterfall approach…



Scrum at a Glance


24 hours
Daily Scrum Meeting
30 days
Backlog tasks expanded by team
Sprint Backlog
Product Backlog
As prioritized by Product Owner
Potentially Shippable Product Increment

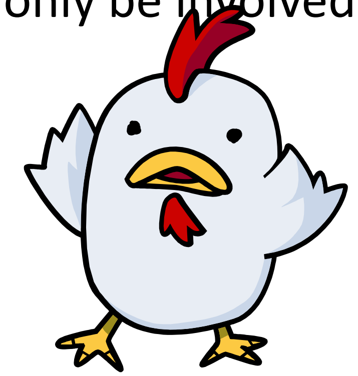


Source: Adapted from *Agile Software Development with Scrum* by Ken Schwaber and Mike Beedle.

"Pigs" and "Chickens"

- **Pig**: Team member committed to success of project
- **Chicken**: Not a pig; interested but not committed

A pig and a chicken are walking down a road. The chicken looks at the pig and says, "Hey, why don't we open a restaurant?" The pig looks back at the chicken and says, "Good idea, what do you want to call it?" The chicken thinks about it and says, "Why don't we call it 'Ham and Eggs'?" "I don't think so," says the pig, "I'd be committed but you'd only be involved."



User Stories

- Instead of Use Cases, Agile project owners do "user stories"
  - **Who** (user role) – Is this a customer, employee, admin, etc.?
  - **What** (goal) – What functionality must be achieved/developed?
  - **Why** (reason) – Why does user want to accomplish this goal?

As a [user role], I want to [goal], so I can [reason].

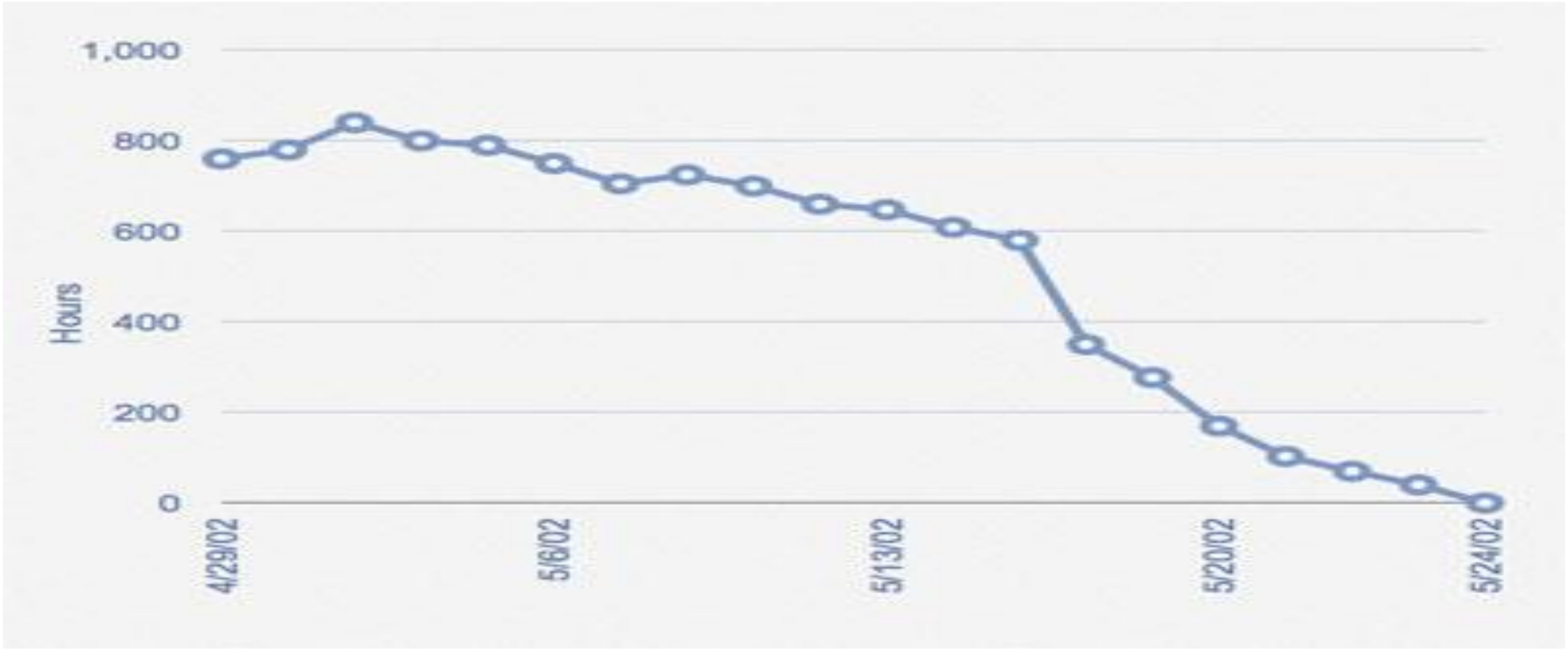
- Example:
  - "As a user, I want to log in, so I can access subscriber content."

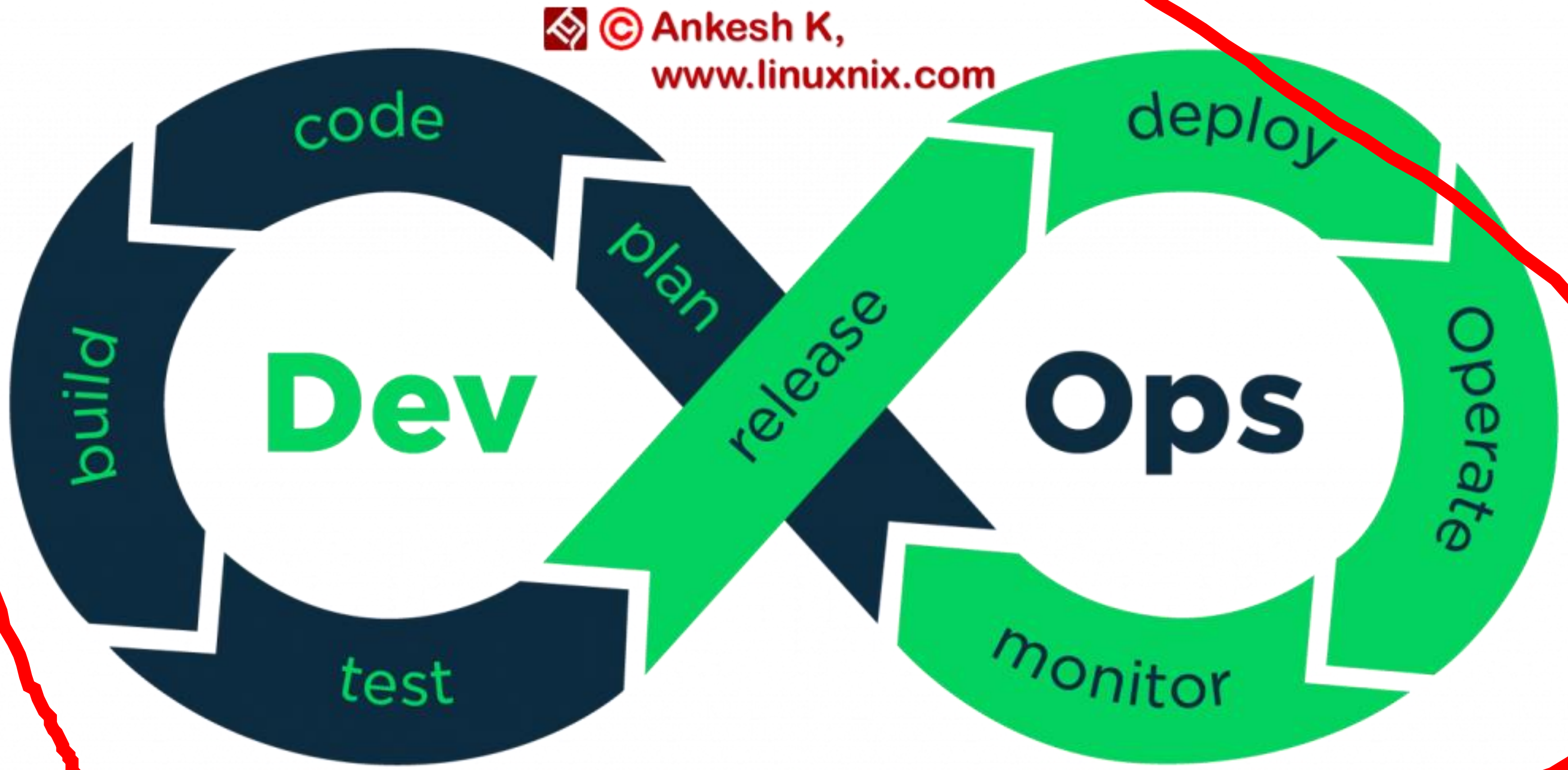
- **story points**: Rating of effort needed to implement this story
  - common scales: 1-10, shirt sizes (XS, S, M, L, XL), etc. especially EFFORT IN HOURS

Sample Product Backlog

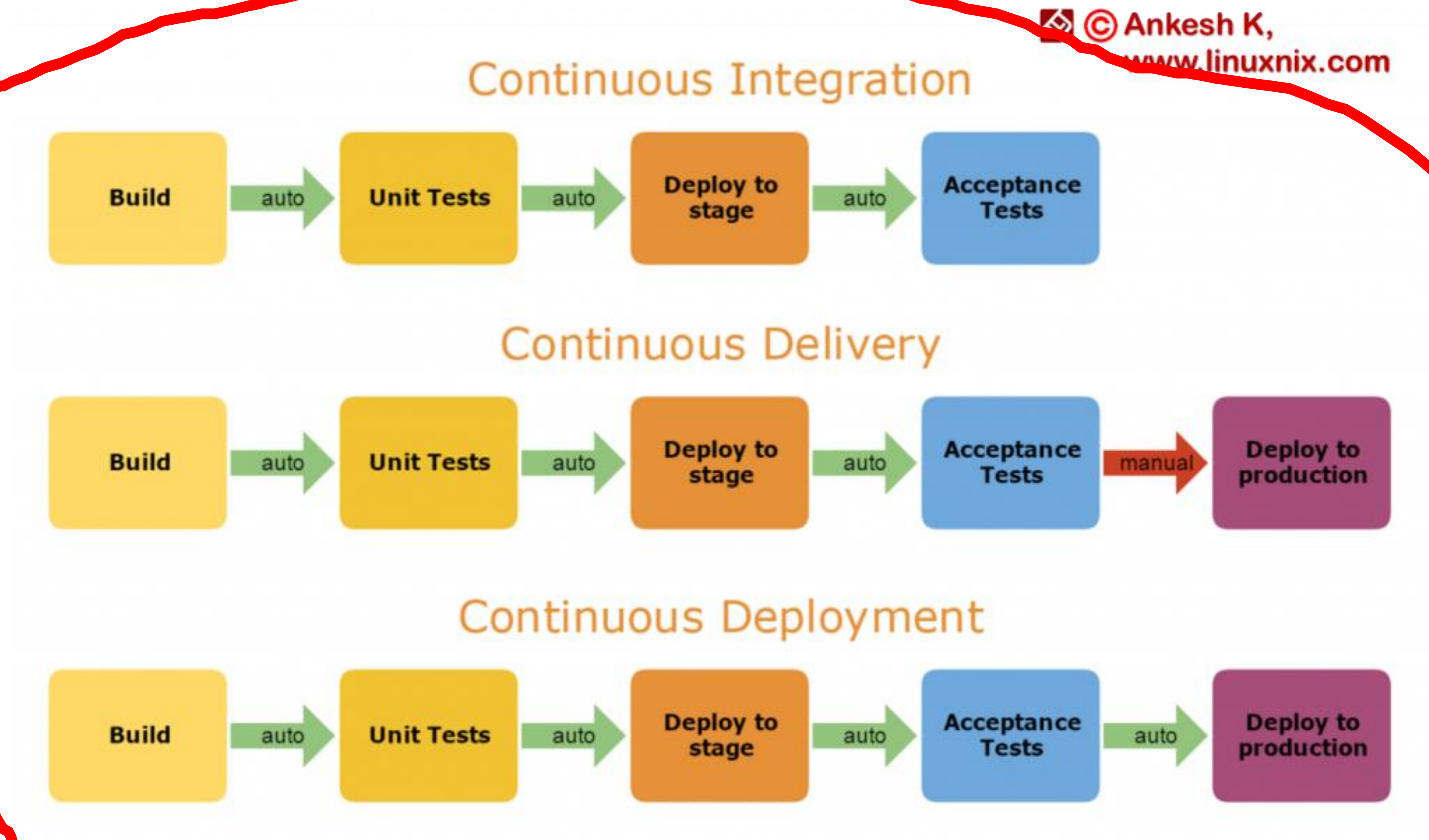
Backlog item	Estimate
Allow a guest to make a reservation	3 (story points)
As a guest, I want to cancel a reservation.	5
As a guest, I want to change the dates of a reservation.	3
As a hotel employee, I can run RevPAR reports (revenue-per-available-room)	8
Improve exception handling	8
...	30
...	50

Burndown Chart





Continuous Integration, Continuous Delivery, Continuous Deployment
And CI-CD Pipeline


© Ankesh K, www.linuxnix.com
Continuous Integration
Build
auto
Unit Tests
auto
Deploy to stage
auto
Acceptance Tests
Continuous Delivery
Build
auto
Unit Tests
auto
Deploy to stage
auto
Acceptance Tests
manual
Deploy to production
Continuous Deployment
Build
auto
Unit Tests
auto
Deploy to stage
auto
Acceptance Tests
auto
Deploy to production

How to model? Too much? Too less?

- Prescriptive/Plan-Driven
- Agile Modeling
- DevOps

Must Read!!!! How to decide life cycle for your project?

- SELECTING A DEVELOPMENT APPROACH
- https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SelectingDevelopmentApproach.pdf

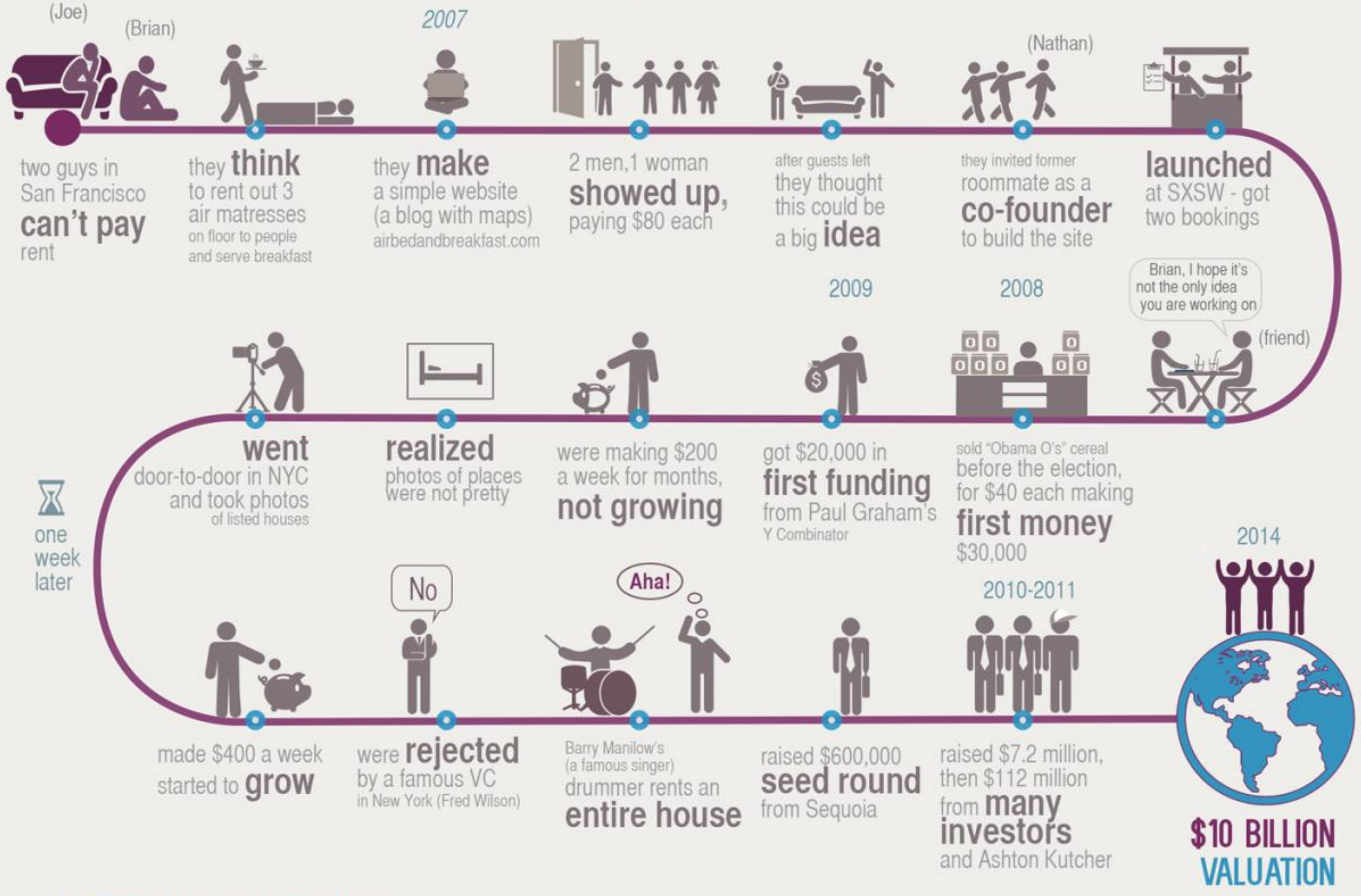
- A closer look at project life cycles
- http://www.ee.co.za/wp-content/uploads/legacy/PositionITSept-Oct%2007-37-44.pdf

Acknowledgement to original creators of images for all pictures in this presentation
Some slides were adapted from David Notkin's course, IIIT-H courses. Thanking them!!!

HOW AIRBNB STARTED

BY ANNA VITAL

Or How 3 Guys Went From Renting Air Mattresses To A 10 Billion Dollar Company



two guys in San Francisco **can't pay** rent

they **think** to rent out 3 air matresses on floor to people and serve breakfast

they **make** a simple website (a blog with maps) airbedandbreakfast.com

2 men,1 woman **showed up,** paying $80 each

after guests left they thought this could be a big **idea**

they invited former roommate as a **co-founder** to build the site

launched at SXSW - got two bookings

sold "Obama O's" cereal before the election, for $40 each making **first money** $30,000

got $20,000 in **first funding** from Paul Graham's Y Combinator

were making $200 a week for months, **not growing**

realized photos of places were not pretty

went door-to-door in NYC and took photos of listed houses

one week later

made $400 a week started to **grow**

were **rejected** by a famous VC in New York (Fred Wilson)

Barry Manilow's (a famous singer) drummer rents an **entire house**

raised $600,000 **seed round** from Sequoia

raised $7.2 million, then $112 million from **many investors** and Ashton Kutcher

$10 BILLION VALUATION

Funders and Founders

based on reports in *Telegraph, WSJ,* and *The Atlantic*